MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

AD A140212

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/CI/NR 84-4D	**2. GOVT ACCESSION NO.**	**3. RECIPIENT'S CATALOG NUMBER**
4. TITLE (and Subtitle) Modelling With Integer Variables		**5. TYPE OF REPORT & PERIOD COVERED** THESIS/DISSERTATION
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James K. Lowe		**8. CONTRACT OR GRANT NUMBER(s)**
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: Georgia Institute of Technology		**10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS**
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433		**12. REPORT DATE** 1984
		13. NUMBER OF PAGES 169
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		**15. SECURITY CLASS. (of this report)** UNCLASS
		15a. DECLASSIFICATION. DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DTIC
ELECTE
APR 1 8 1984
S
A

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17

13 april 84

LYNN E. WOLAVER
Dean for Research and
Professional Development
AFIT, Wright-Patterson AFB OH

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ATTACHED

84 04 16 037

MODELLING WITH INTEGER VARIABLES

A DISSERTATION

by

James Kenneth Lowe

Presented to
The Faculty of the Division of Graduate Studies

In Partial Fulfillment
of the Requirements for the Degree
Ph.D. in the College of Management

Georgia Institute of Technology
February 1984

84 04 16

Abstract

Modelling With Integer Variables

Capt. James K. Lowe, USAF

Ph.D. Management, 1984

Directed by Dr. R. G. Jeroslow

Georgia Institute of Technology

169 pages

Representing nonlinear optimization problems as mixed-integer programs has largely been considered as; 1) an "art" with few unresolved theoretical issues, and 2) a fairly standard "preprocessing" routine when combined with several ad hoc modelling improvements which have evolved through computational experience. The more common avenue of research in mixed-integer programming has focused upon finding improved algorithms and heuristics to solve the problems, assuming a standard mixed-integer representation exists. In this thesis, we take a "step backwards" and re-examine the theoretical issues and subtleties involved in representing problems with mixed-integer representations.

As a result of our theoretical investigation, necessary and sufficient conditions are given which guarantee the existence of a mixed integer representation for a given problem. The conditions explain some of the subtleties surrounding well-known problems such as the fixed-charge problem, which requires an upper bound on the variable. Our results, always in the rational field, concern the representability of a finite union of sets. Many results extend Rockafellar's finite basis theorems.

In our theoretical investigation, we employ a system of inequalities derived from disjunctive programming to help prove the existence of a mixed-integer representation for a finite union of polyhedra whose recession directions satisfy certain criteria. While the system of inequalities is not new, its use as a mixed-integer program representation is new. Another mixed-integer representation is presented which utilizes the "extreme points" of the individual polyhedra (when representing a finite union of polyhedra). Both representations are automatic, easy to use, and both have the property of sharpness. Sharpness concerns the amount of information lost as the integer variables are relaxed to the continuous rational field.

Whether a particular representation is sharp or not often determines whether a problem is solvable within minutes of computation or remains unsolved after hours of computation. Our automatic representations are always sharp, they often preserve their sharpness after variable arbitration (as in a branch-and-bound setting), and have shown tremendous efficiency and time savings for the problems and modellings explored.

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to ascertain the value and/or contribution of research accomplished by students or faculty of the Air Force Institute of Technology (ATC). It would be greatly appreciated if you would complete the following questionnaire and return it to:

AFIT/NR
Wright-Patterson AFB OH 45433

RESEARCH TITLE: __Modelling With Integer Variables__

AUTHOR: __James K. Lowe__

RESEARCH ASSESSMENT QUESTIONS:

1. Did this research contribute to a current Air Force project?

( ) a. YES   ( ) b. NO

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not?

( ) a. YES   ( ) b. NO

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency achieved/received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of manpower and/or dollars?

( ) a. MAN-YEARS _____   ( ) b. $_____

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3. above), what is your estimate of its significance?

( ) a. HIGHLY   ( ) b. SIGNIFICANT   ( ) c. SLIGHTLY   ( ) d. OF NO
       SIGNIFICANT   SIGNIFICANT   SIGNIFICANCE

5. AFIT welcomes any further comments you may have on the above questions, or any additional details concerning the current application, future potential, or other value of this research. Please use the bottom part of this questionnaire for your statement(s).

NAME   GRADE   POSITION

ORGANIZATION   LOCATION

STATEMENT(s):

AFIT/ NR
WRIGHT-PATTERSON AFB OH 45433
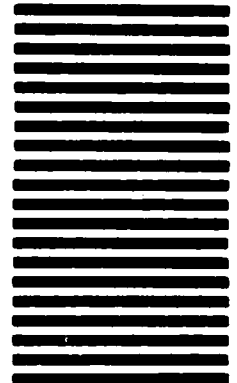
OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE. $300

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS   PERMIT NO. 73236   WASHINGTON D.C.

POSTAGE WILL BE PAID BY ADDRESSEE

AFIT/ DAA
Wright-Patterson AFB OH 45433

TABLE OF CONTENTS

ACKNOWLEDGMENTS

CHAPTER I

INTRODUCTION

A widely accepted view in integer programming research is that
progress, in utilizing (mixed-) integer formulations to solve real-world
problems, will rely primarily on algorithm advances (for general and
special structures) and clever coding tricks. In this view, we are
perceived as already knowing how to represent a real-world problem with
integer variables; those simple "formulation techniques" appear early in
the subject and by now are widespread in the masters'-level and even
undergraduate-level textbooks. (see e.g. [14], [42]) However, since in
the 1970's, the experience of practitioners indicate that some major
issues of formulation have been overlooked. For example, Geoffrion and
Graves [16] solve a large scale multicommodity distribution problem which
includes fixed charges. In their modelling, they notice that to
economize on the number of constraints, a standard linear programming
technique, results in more iterations for convergence to the optimal
solution. In another example, Williams [41] finds a similar advantage of
disaggregating constraints for logical system problems, which he models
as generalized set-covering problems. Both of the above references
indicate a computational need to reformulate MIP representations before
attempting to solve them.

One of the earliest integer models is the Fixed-Charge problem in which a fixed cost is incurred (the objective is to minimize costs) when a certain activity level is non-zero (see Dantzig [10]). This very common problem can be modelled as a mixed-integer program (MIP), provided that an explicit upper bound is known for the variable which "triggers" the fixed cost. A seemingly similar problem is the Fixed-Benefit problem, in which a benefit (rebate) is received, rather than a cost incurred when the variable is nonzero. This problem cannot be represented as a MIP even <u>with</u> a known upper bound for the "trigger" variable. As we shall see in Chapter 2, the Fixed-Benefit case requires an explicit minimum usage level which must be met before the benefit is received, if it is to be represented via (bounded) integer variables.

In contrasting the fixed-charge and fixed-benefit problems, we have illustrated one of the <u>subtleties</u> of modelling with integer variables that we will treat in this thesis. Several other aspects of integer modelling will be discussed, and we present here (we hope convincing) evidence of our two theses, namely that: 1) There are substantial advantages to using "better" integer models; and 2) The study of integer modelling has many aspects which are amenable to exact, analytic development.

In the late sixties, and early seventies, Benichou, et al. [6], Land and Doig, Beale, and others implement improved Branch-and-Bound based commercial mixed-integer programming systems capable of solving many practical industrial MIPs [41]. These algorithmic advances provide opportunities to conduct computational experiments to compare various

ad hoc modellings and reformulations. The results of these experiments, notably those of H. P. Williams, and Geoffrion and Graves (who used a specially-designed enumerative code) motivate others to re-examine MIP modelling issues for computational improvements.

Williams [41] finds, in comparing modellings, that "the superior formulation is always the one which is "tighter" in a continuous sense." By "tighter," he means the continuous optimum (integrality relaxed) is closer to the integer optimum. Williams experiments involve integer variables in both flow problems and logic problems. His ad hoc methods of problem reformulation involve coefficient reduction and disaggregation of constraints.

Geoffrion and Graves [16] model a multicommodity distribution system and provide a "lesson on Model Representation." (see Section 5 of [16].) In their flow problem, they provide computational results favoring a modelling which disaggregates constraints. They realize that the disaggregation, in the relaxed case, provides the convex hull of the original integer feasible solutions. They also note that the price of the tighter bound and reduction in Branch-and-Bound tree branching is the additional time required to solve the larger LP relaxation at each node of the Branch-and-Bound tree. Based upon their computational and theoretical results, they "suggest a general methodology for discovering improved model representations: for various subsets of contraints involving some of the integer variables, try to explicitly derive the convex hull of the integer feasible points." Much of our work in this

thesis is in effect a partial reply to their request for such a
methodology.

Rardin and Choe [40] disaggregate relevant constraints of a fixed-
charge multicommodity network flow problem by way of an arc-node
representation which explictly defines each possible path from each
source i, to all sinks j. While improving the value of the LP
relaxation, this technique often greatly increases the size of the
resulting MIP. They provide computational results, supporting the
effectiveness of the improved LP relaxations.

Oley and Sjoquist [38], Crowder, Johnson, and Padberg [9],
Johnson, Kostreva, and Suhl [28] automate the process of coefficient
reduction and constraint disaggregation. Oley and Sjoquist implement
their reformulation techniques in CDC's APEX IV math programming system.
They employ a pruning technique similar to that found in Chapter V of
this thesis, and a "big M" reduction method similar to that employed in
Chapter II of this thesis.

Johnson, et al. [28] employ similar reductions and disaggregations
in a large scale planning scenario. In their problem, they encounter
fixed-charges, either/or type constraints, and special ordered set (SOS)
variables. They do not disaggregate the either/or type constraints as in
Chapter II of this thesis.

The reformulation sources mentioned above appeal to the
"computational effectiveness" justification of their techniques. While
their results are extremely useful, they do not systematically address
the issue of existence of a MIP modelling for a given problem, nor

techniques for providing modellings with tight linear relaxations, nor the "hereditary" effects of various formulations as branching proceeds in a branch-and-bound procedure.

As we shall see later in this thesis, improvements in modelling techniques derive from what begins as theoretical investigations. In the mid seventies, Ibaraki [21], and Meyer [33], [34], [35] launch a systematic study of modelling problems as MIPs. In accounting for their earlier results on the existence of modellings, our proofs in Chapter II will amount to formulation techniques which have, and hereditarily preserve, tight linear relaxations. Our results are either derived from, or motivated by disjunctive programming. In later chapters of the thesis, we report experiments on the new formulation techniques.

As in our approach, Balas [1] also expresses the discrete optimization problem as the intersection of unions of polyhedra. In that paper, he introduces operations to reduce the number of intersections, which strengthens the relaxations. In this manner one creates a hierarchy of relaxations ranging from the original LP relaxation to the exact convex hull of integer feasible points. This hierarchy of relaxations in some respects goes beyond the theoretical work reported here, although it is similar to the lattice of co-propositions in [24] that contains several hierarchies.

In Chapter II of this thesis, we develop necessary and sufficient conditions concerning when a union of polyhedra is MIP representable. In doing so, we develop an automatic MIP representation for any bounded-MIP representable union of polyhedra.

We then focus upon the sharpness of this MIP representation.
Sharpness holds when the LP relaxation of a MIP modelling is <u>exactly</u> the
convex hull of integer feasible solutions. Our automatic MIP
representation is always sharp. Typically, it is obtained for a <u>subpart</u>
of the whole mixed-integer program, such as a fixed-charge, piecewise-
linear function, either/or constraints, etc. This part must be then
"linked" into the main program, and typically the result is not sharp for
the whole program. The issues raised by the "linkage" process are in
part the substance of Balas' hierarchy in [1].

In Chapter III we review bounded-MIP representability and conduct
two experiments comparing our "sharp" representations to common
representations found in current literature use. Experiments are
necessary since the SHARP representation (which subsumes the
disaggregation techniques employed by others) often, but not always,
results in much larger problems in terms of both constraints and
continuous variables.

The first experiment compares our larger sharp MIP modelling of
"either/or" type constraints with a more concise, non-sharp modelling
found in many current articles and textbooks ([14] for instance). The
dramatic lack of sharpness of the common modelling proves to be its major
defect, as the sharp modelling performs faster in terms of cpu seconds
and more efficiently in terms of Branch-and-Bound nodes solved (many
instances show an improvement of over 100%, increasing with problem
size). From our results, only in very small instances did the "non-

sharp" model perform faster than our "sharp" representation. As others have found using ad hoc techniques, we find "sharpness" as a key ingredient to successful integer programs.

The second experiment in Chapter III, tests whether it is computationally favorable to disaggregate a non-linear function into "easier" components; model each component individually; and then "link" the models together by termwise addition in the objective function before solving the final MIP. We test this aspect of modelling linkage on a piecewise-linear function with fixed-charges. The results favor the modelling of the entire function, even when the "linked" modelling parts are modelled using our sharp modelling.

Besides testing modelling linkage, this experiment also provides evidence favoring "hereditarily" sharp models. Unless some variable co-ordination is performed upon the linked models, they rarely will be hereditarily sharp. Thus, once the Branch-and-Bound algorithm begins to arbitrate variables, the modelling loses it sharpness, resulting in many more branch-and-bound nodes.

In Chapter IV we provide computational support of a common conjecture that the proximity of the convex hull of the integer feasible solutions to the set of integer feasible solutions is a guide towards problem difficulty. For this experiment, we use the Fixed-Charge, and Fixed-Benefit problems mentioned earlier. The Fixed-Charge problem is a very well-known difficult mixed-integer problem. On the other hand, it turns out that the Fixed-Benefit problem is relatively easy to solve as a

MIP. The Fixed-Charge problem usually has a large gap between the two sets mentioned above, while the Fixed-Benefit problem with low usage levels (as common in practice) has a much smaller, nearly indistinguishable difference between the set of integer feasible points and its corresponding convex hull. Our computational results support the stated conjecture, as the Fixed-Charge problem is much more difficult than the Fixed-Benefit problem.

In Chapter V, we include two experiments involving the modelling of propositional logic problems as mixed-integer programs. We find these problems extremely easy to solve using elementary modelling techniques that have been standard since the 1960's. Also, we can show an advantage of our modelling techniques over the standard ones.

In summary, we have not only developed theoretical conditions concerning the existence of MIP representations, but have, in doing so, developed an automatic modelling for every bounded-MIP representable problem instance. The modelling developed is always sharp, a property which we, and others, have found to greatly improve computational performance in a Branch-and-Bound mixed-integer programming system. We provide several experiments to test various concepts and properties of our automatic modellings, and MIP modellings in general. We find Sharpness and hereditary sharpness as essential properties for successful model building. The fact that our modelling is completely automatic, and always sharp, greatly enhances its use as a modelling technique.

REFERENCES

1. E. Balas, "Disjunctive Programming and a Hierarchy of Relaxations for Discrete Optimization Problems," Carnegie-Mellon tech report, June 1983.

2. E. Balas, "Disjunctive Programming: Cutting-planes from Logical Conditions," in O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Nonlinear Programming 2, Academic Press, New York (1975), pp. 279-312.

3. E. Balas, "Disjunctive Programming: Facets of the Convex Hull of Feasible Points," no. 348, GSIA, Carnegie-Mellon University, 1974.

4. E. Balas and M. W. Padberg, "Set Partitioning: A Survey," SIAM Review 18 (1976), pp. 710-760.

5. Bazaraa, M. and Shetty, M., Non-Linear Programming, John Wiley & Sons, Inc., New York 1979.

6. Beale, E. M. L., "Branch-and-Bound Methods for Mathematical Programming," in Discrete Optimization II, eds. P. L. Hammer, E. L. Johnson, B. H. Korte, North-Holland Publishing Co., Amsterdam, pp. 201-221, 1979.

7. E. M. L. Beale and J. J. H. Forrest, "Global Optimization Using Special Ordered Sets," Mathematical Programming 10 (1976), pp. 52-69.

8. M. Benichou, J.-M. Gauthier, G. Hentges, G. Ribiere, "The Efficient Solution of Large-Scale Linear Programming Problems-Some Algorithmic Techniques and Computational Results," Mathematical Programming 13 (1977), pp. 280-322.

9. Bixby, Robert E., Matroids and Operations Research, Northwestern University, Illinois, 1984.

10. C. E. Blair and R. G. Jeroslow, "A Converse for Disjunctive Constraints," Journal of Optimization Theory and Its Applications 25 (1978), pp. 195-206.

11. S. Bradley, A. Hax and T. Magnanti, Applied Mathematical Programming, Addison-Wesley Pub. Co., Reading, Mass., 1977.

12. H. Crowder and E. L. Johnson, "Solving Large-Scale Zero-One Linear Programming Problems," Operations Research (1983), pp. 803-834.

13. G. B. Dantzig, Linear Programming and Extensions, Princeton, New Jersey, Princeton University Press, 1963.

14. R. Davis and D. Lenat, _Knowledge-Based Systems in Artificial Intelligence_, McGraw-Hill Book Co., New York, 1982.

15. R. Davis and B. Buchanan, "Production Rules as a Representation for a Knowledge-Based Consultation Program," _Artificial Intelligience_ 8, (1977), pp. 15-45.

16. J. Edmonds, "Matroids and the Greedy Algorithm," _Mathematical Programming_ 1 (1971), pp. 127-136.

17. G. D. Eppen and F. J. Gould, _Quantitative Concepts for Management_, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1979.

18. J.-M. Gauthier and G. Ribiere, "Experiments in Mixed-Integer Linear Programming Using Pseudo-Costs," _Mathematical Programming_ 12 (1977), pp. 26-47.

19. A. M. Geoffrion and G. W. Graves, "Multicommodity Distribution System Design by Benders Decomposition," _Management Science_ 20 (1974), pp. 822-844.

20. F. Glover, "New Results on Equivalent Integer Programming Formulations," _Mathematical Programming_ 8 (1975), pp. 84-90.

21. F. Glover, "Polyhedral Annexation In Mixed Integer and Combinatorial Programming," _Mathematical Programming_ 9 (1975), pp. 161-188.

22. R. E. Gomory, "An Algorithm for Integer Solutions to Linear Programs," in R. L. Graves and P. Wolfe, ed., _Recent Advances in Mathematical Programming_, McGraw-Hill, 1983.

23. A. C. Ho, "Cutting-planes for Disjunctive Programs: Balas' Aggregated Problem" Carnegie-Mellon University, October 1976 (a Ph. D. student summer research paper).

24. T. Ibaraki, "Integer Programming Formulation of Combinatorial Optimization Problems," _Discrete Mathematics_ 16 (1976), pp. 39-52.

25. R. H. F. Jackson and R.P. O'Neill, eds. _COAL Special Issue: Mixed Integer Programming in Mathematical Programming Systems_, an ORSA and COAL/MPS publication.

26. R. Jeroslow, "Cutting-plane Theory: Disjunctive Methods," _Annals of Discrete Mathematics_ 1 (1977), pp. 293-330.

27. R. Jeroslow, "Cutting-planes for Relaxations of Integer Programs," no. 347, GSIA, Carnegie-Mellon University, 1974.

28. R. Jeroslow, "Representations of Unbounded Optimizations as Integer Programs," *Journal on Optimization Theory and Its Applications* 30 (1980), pp. 339-351.

29. R. G. Jeroslow and J. K. Lowe, "Modelling with Integer Variables," Ga. Tech report (rev.), March 1983.

30. R. G. Jeroslow and J. K. Lowe, "Experimental Results on the New Techniques for Integer Programming Formulations," Ga. Tech report, June 1983.

31. E. L. Johnson, M. M. Kostreva, and U. Suhl, "Solving 0-1 Integer Programming Problems Arising from Large Scale Planning Models," IBM report RC9349, April 1982.

32. C. B. Krabek, "Some Experiments in Mixed Integer Matrix Reduction," Control Data Corporation, presented at ORSA/TIMS Hawaii Meeting, 1979.

33. A Land and S. Powell, *Fortran Codes for Mathematical Programming: Linear, Quadratic and Discrete*, John Wiley & Sons, London, England, 1973.

34. A. Land and S. Powell, "A Survey of Available Computer Codes to Solve Integer Linear Programming Problems," Research Report No. 81-9, London School of Economics and Political Science, London, April 1981.

35. E. Mendelson, *Introduction to Mathematical Logic*, D. Van Nostrand Co. Inc., Princeton.

36. R. R. Meyer, "Integer and Mixed Integer Programming Models: General Properties," *Journal on Optimization Theory and its Applications* 16 (1975), pp. 191-206.

37. R. R. Meyer, "Mixed-Integer Minimization Models for Piecewise-Linear Functions of a Single Variable," *Discrete Mathematics* 16 (1976), pp. 163-171.

38. R. R. Meyer, "A Theoretical and Computational Comparison of 'Equivalent' Mixed Integer Formulations," *Naval Research Logistics Quarterly* 28 (1981), pp. 115-131.

39. R. R. Meyer and M. V. Thakkar, "Rational Mixed Integer Minimization Models," MRC #1552, University of Wisconsin, 1976.

40. G. Mitra, "Investigation of Some Branch-and-Bound Strategies for the Solution of Mixed-integer Linear Programs," *Mathematical Programming* 4 (1973), pp. 155-170.

41. L. A. Oley and R. J. Sjoquist, "Automatic Reformulation of Mixed and Pure Integer Models to Reduce Solution Time in Apex IV," Control Data Corporation, presented at ORSA/TIMS San Diego Meeting October 1982.

42. R. Gary Parker and R. L. Rardin, Discrete Optimization, (in print) Ga. Inst. of Technology, Atlanta, 1984.

43. R. L. Rardin and U. Choe, "Tighter Relaxation of Fixed Charge Network Flow Problems," Report #J-79-18, Ga. Inst. of Technology, Atlanta, May 1979.

44. Rockafellar, R. T., Convex Analysis, Princeton University Press, 1970.

45. D. C. Sommer, "Computational Experience with the Ophelie Mixed Integer Code," Control Data Corporation, presented at ORSA/TIMS, 1970.

46. Stoer, J., and Witzgall, C., Convexity and Optimization in Finite Deimensions I, Springer-Verlag, New York, 1970.

47. H. P. Williams, "Experiments in the Formulation of Integer Programming Problems," Mathematical Programming Study 2, 1974, pp. 180-197.

48. H. P. Williams, Model Building in Mathematical Programming, John Wiley and Sons (Wiley Interscience), 1978.

CHAPTER II

MODELLING WITH INTEGER VARIABLES

Introduction

The interest in modelling practical and mathematical problems as integer and mixed-integer programs begins with a paper by Dantzig [4]. Since that paper, many "standard problem formulations" are routinely available in elementary textbooks. However, not until the more recent work of Meyer ([9], [10] , [11] has there been thorough exploration, or, perhaps, even awareness, of the "limits" of integer modelling. Our aim in this chapter is to elucidate some of the subtleties of integer modelling.

For example, the fixed-charged problem can be modelled without a minimum usage constraint. On the other hand, the "fixed-benefit" problem must contain a minimum usage stipulation (which must be met before the benefit is received) before it can be modelled as an integer problem. Our results explain why this is so.

In section 1 of this chapter, we find that, for bounded sets, or functions defined on bounded domains (Lemma 2.1.2), the subtleties are neither very complex, nor restrictive. However, for a few unbounded sets the restrictions are so severe that, unless the unboundedness can be removed, one would naturally seek a different mathematical formulation of the sets. For example, many unbounded functions can be modelled as

generalized linear complementarity problems, which have no integer
modelling (the set $\{x.y) \geqslant 0 \mid x \cdot y = 0\}$ is one such; see Theorem 2.1.7).

Even with the unboundedness removed, it is important to know that
a bound must be used. This kind of information occurred in Meyer's
earliest studies of integer modelling [9] in connection with the fixed-
charge problem (which is representable only in its bounded form, when
rational data are stipulated). This choice of a suitable bound can be a
major issue in algorithmic implementation.

In section 1, we develop basic definitions and many corollaries
which lead to the heart of our investigations into set representability.
Theorem 2.1.7 describes necessary and sufficient conditions for the union
of representable sets to be MIP-representable. In section 2, we explore
Meyer's concept of bounded representability, and develop necessary and
sufficient conditions for bounded MIP-representability (Theorem 2.2.1).
For bounded representable sets, our results guarantee one representation
in which all integer variables are binary variables; furthermore, these
binary variables occur in a common set-partitioning constraint.

We then focus upon the implementation of MIP-representations, and
ask how "accurate" the representations are after the integer variables
are "relaxed" to continuous (rational) variables, as in branch-and-bound
algorithms (section 3). We easily show that the best possible relaxed
MIP-representation contains the convex hull of the original set. When
the specific MIP-representation is exactly the convex hull, it is called
a "sharp" representation. We provide sharp representations for many
models; most representations in common use are sharp.

In our last section of this chapter (section 4), we show that Meyer's bounded-MIP representability Theorems [9] are specific cases of our Theorem 2.1.7 when all variables are restricted to the rational field. We then apply a linear inequality MIP-representation for polyhedra developed in section 1 to an example representation used by Beale [3], and show that the two can be used interchangeably and that both are "sharp" representations. Moreover, it typically occurs, when we turn our representability techniques to problems treated earlier, that we obtain the most efficient (fewest constraint) linear relaxation, even while our relaxation is best possible (i.e. "sharp"). To do so may require noting some algebraic simplifications for the specific problem, but that is all. Our problem formulations are either derived from or motivated by disjunctive programming [1], [7].

While Meyer's theory permits representations which involve irrational data, we have restricted ourselves to rational representations only. This appears to cover all that is of practical interest, and so our results are stated for the rational field only. In some cases, the results extend to the real field, but such issues can be addressed later if they seem of interest.

Throughout this thesis, we denote the convex span respectively the closure of set S as conv(S) respectively cl(S), and clconv(S) is the closure of conv(S) [12]. Similarily cone (S) denotes the convex cone generated by S. The convex span and cone operations are taken only with rational multipliers.

We assume that the reader is familiar with the intended use of
MIP-representability in problem formulations, as illustrated in Meyer's
papers.

Section 1: Representability of the Finite Union of Representable Sets

We define a set $S \subseteq Q$ (where Q denotes the rationals) as MIP-
representable if there are rational matrices A_1, A_2, A_3, and vector b
with the property that: $x \in S$, if and only if, for some u, v $\geqslant$ 0, with
u integer, we have

$$A_1 x + A_2 u + A_3 v = b. \qquad (2.1.1)$$

We note that our definition extends to the case where x is constrained to
be integer. I.e., let $x = (x_1, \ldots, x_n)$ and let I be any subset of
$\{1, \ldots, n\}$; then if S is MIP-representable so is the set $\{x \in S | x_i \in Z$
(integers) for all $i \in I\}$ (this is an easy exercise). An extension of
our definition of MIP-representation is to functions: f is MIP-
representable if the set epi(f) is; where epi(f) = $\{(z,x) | z \geqslant f(x)$ and
f(x) is defined} denotes the epigraph of f. We also note that f has a
rational MIMM on T in Meyer's sense [10] and only if f is MIP-
representable in the above sense.

Our first theorem is an extension of the finite basis theorem for
polyhedra (Rockafellar [12]).

Theorem 2.1.1

Suppose that $S \neq \emptyset$ is MIP representable. There are constants a,

$b > 0$ such that for all integers $p > 1$, $S_{a+bp} \neq \emptyset$ and

$$S = S_{a+bp} + pM, \qquad (2.1.2)$$

where

$$S_\alpha = \{x \in S \mid \sum_{i=1}^{n} | x_i | < \alpha\}, \text{ and} \qquad (2.1.3)$$

$$M = \{x \in Z^n \mid \text{ for some } u, v > 0, \text{ with } u, v \text{ integer}, \qquad (2.1.4)$$
$$A_1 x + A_2 u + A_3 v = 0\}.$$

Proof

The containment $S \supseteq S_{a+bp} + pM$ is immediate once we show that $S_{a+bp} \neq \emptyset$. In fact, if $x \in S$ and $x' \in M$ then by a direct computation, $x + px' \in S$.

We note that the nonempty set P, defined as

$$P = \{(x,u,v) \mid A_1 x + A_2 u + A_3 v = b; u, v > 0\} \qquad (2.1.5)$$

is a polyhedron. Therefore, from the finite basis theorem, there are disjoint finite sets J and K and points (x^t, u^t, v^t) of P for $t \in K$ with

$$P = \text{conv}(\{(x^j, u^j, v^j) \mid j \in J\}) + \qquad (2.1.6)$$
$$\text{cone}(\{(x^k, u^k, v^k) \mid k \in K\}).$$

We define $a' = \max |x^j|$, and $b = \sum_{k \in K} |x^k|$, and $a = \max\{a', |x^0|\}$ for $x^0 \in S$.

(Note, $|x| = |x_1| + |x_2| + \ldots + |x_n|$ if $x = (x_1, x_2, \ldots, x_n)$). Since $b \geqslant 0$ for $p \geqslant 1$ we have $x^0 \in S_{a+pb}$. Thus $S_{a+bp} \neq \emptyset$ and we now must establish the containment $S \subseteq S_{a+bp} + pM$.

We may assume in what follows, that (x^k, u^k, v^k) is an integer vector for $k \in K$ (recall that A_1, A_2, A_3, b are rational). In particular, $x^k \in M$ for $k \in K$.

Let $x \in S$. Then for some u, $v \geqslant 0$ with u integer, we have $(x,u,v) \in P$. Therefore there is a solution to

$$(x,u,v) = \sum_{j \in J} \lambda_j (x^j, u^j, v^j) + \sum_{k \in K} \tau_k \, p(x^k, u^k, v^k), \qquad (2.1.7)$$

$$1 = \sum_{j \in J} \lambda_j \; ; \; \lambda_j \geqslant 0, \; j \in J, \; \tau_k \geqslant 0, \; k \in K.$$

Let q_k denote the integer part of τ_k and let f_k be the fractional part, so that $\tau_k = q_k + f_k$, $f_k < 1$, and q_k, $f_k \geqslant 0$. Define

$$(x', u, v') = \sum_{j \in J} \lambda_j (x^j, u^j, v^j) + \sum_{k \in K} f_k p(x^k, u^k, v^k) \qquad (2.1.8a)$$

$$(x^*, u^*, v^*) = \sum_{k \in K} q_k p(x^k, u^k, v^k). \text{ From (2.1.8a),} \qquad (2.1.8b)$$

so we have $x = x' + x^*$. We must show that $x' \in S_{a+bp}$ and $x^* \in pM$.

Since $(x,u,v) = (x', u', v') + (x^*, u^*, v^*)$ and u and u^* are

integer vectors, u' is an integer vector. From (2.1.8a), $(x', u', v') \in$
P. Hence $x' \in S$. Also trom (2.1.5), and (2.1.8a),

$$|x'| < a' + p \sum_{k \in K} f_k |x^k| < a' + pb. \text{ Thus } x' \in S_{a+bp}. \text{ Since } x^k \in M \text{ and } q_k$$

is integer for $k \in K$, we have $x^* \in pM$ from (1.8b).

<div align="right">Q.E.D.</div>

We note that 2.1.1 defines any representable set S in terms of a
bounded part (S_{a+bp}) and a discrete unbounded part (pM) as in the finite
basis theorem for polyhedra. For $\alpha \in Q$, S_α is MIP-representable; hence
S_{a+bp} is representable. In addition, M is an integral monoid with a
finite basis (see Jeroslow [8]).

Corollary 2.1.2

If S is both bounded and MIP-representable, S is a finite union of
polytopes.

Proof

It suffices to show that a bound can be placed on the vector u
occuring in the representation.

Assume that $x \in S$. From the proof of Theorem 2.1.1, and since S
has no recession directions, we have $x^k = 0$ for $k \in K$. Hence in
(2.1.8a), $x' = x$ and $A_1 x + A_2 u' + A_3 v' = 0$ with u', $v' \geq 0$ and u'
integer. (recall $x \in S$) A bound on $|u'|$ is $\max_{j \in J} |u^j| + \sum_{k \in K} |u^k|$ directly
from (2.1.8).

<div align="right">Q.E.D.</div>

Corollary 2.1.3

If S is representable, conv(S) is a polyhedron.

Proof

Apply (2.1.2) for p=1. Since M is a monoid we have

$$\text{conv}(S_{a+b} + M) = \text{conv}(S_{a+b}) + \text{cone}(M). \qquad (2.1.9)$$

To establish (2.1.9), note that clearly

$$\text{conv}(S_{a+b} + M) \subseteq \text{conv}(S_{a+b}) + \text{cone}(M) \qquad (2.1.10)$$

is valid. To prove that

$$\text{conv}(S_{a+b} + M) \supseteq \text{conv}(S_{a+b}) + \text{cone}(M), \qquad (2.1.11)$$

we have, for any $v \in \text{conv}(S_{a+b}) + \text{cone}(M)$,

$$v = \sum_j \lambda_j s^j + \sum_k u_k m_k, \qquad (2.1.12)$$

where $\sum_j \lambda_j = 1$, $\lambda_j > 0$, $u_k > 0$, $m_k \in M$, and $s^j \in S_{a+b}$.

let $\lambda_1 > 0$ (WLOG), then we have

$$m = \sum_k u_k m_k / \lambda_1 \in \text{cone}(M), \text{ and also} \qquad (2.1.13)$$

$$v = \lambda_1(s' + m) + \sum_{j \neq 1} \lambda_j(s^j + 0). \qquad (2.1.14)$$

But since $(s' + m) \in S_{a+b} + \text{cone}(M)$, then $v \in \text{conv}(S_{a+b} + \text{cone}(M))$.

Furthermore, if $w = s + w'$, where $w' \in \text{cone}(M)$ and $s \in S_{a+b}$, then for

some integer $D \geq 1$, $Dw' \in M$, and by factoring,

$$s + w' = 1/D(s + Dw') + 1/D(s+0) + \ldots + 1/D(s+0) \qquad (2.1.15)$$

where $(s + Dw') \in S_{a+b} + M$ and $(s +0) \in S_{a+b} + M$. Therefore

$(s + w') \in \text{conv}(S_{a+b} + M)$, and

$$S_{a+b} + \text{cone}(M) \subseteq \text{conv}(S_{a+b} + M). \qquad (2.1.16)$$

Finally,

$$\text{conv}(S_{a+b}) + \text{cone}(M) \subseteq \text{conv}(S_{a+b} + \text{cone}(M)) \subseteq \text{conv}(S_{a+b} + M). \qquad (2.1.17)$$

which proves (2.1.9)

By Corollary 2.1.2, $\text{conv}(S_{a+b})$ is a polytope. Since M has a

finite basis, $\text{cone}(M)$ is a polyhedral cone. By the converse to the

Finite Basis Theorem (Rockafellar [11]), $\text{conv}(S)$ is a polyhedron.

Q.E.D.

Corollary 2.1.4

S is an MIP-representable set if and only if there is a bounded MIP-representable set S' and an integral monoid M with a finite basis such that $S = S' + M$.

Moreover, if S' is an MIP-representable set (bounded or not) and M is an integral monoid with finite basis, then $S = S' + M$ is MIP-representable.

Proof

The "only if" is established by Theorem 2.1.1.

If $S = S' + M$, let A_1, A_2, A_3, and b be rational matrices and vector such that

$$\text{if } x \in S' \leftrightarrow \text{ for some } u, v \geqslant 0, u \text{ integer}, \qquad (2.1.18)$$
$$A_1 x + A_2 u + A_3 v = b,$$

and let A_4 be an integral matrix such that

$$m \in M \leftrightarrow \text{ for some } w \geqslant 0 \text{ integer} \qquad (2.1.19)$$
$$m = A_4 w.$$

Then we have,

$$x \in S \leftrightarrow \text{ there are } x_1 \text{ and } x_2, u, v, w \geqslant 0 \text{ with } u, w \text{ integer} \qquad (2.1.20)$$
$$\text{such that } x = x_1 + x_2, A_1 x_1 + A_2 u + A_3 v = b, \text{ and}$$
$$A_4 w = x_2,$$

which is an MIP-representation of S.

<div align="right">Q.E.D.</div>

The next result is a technical corollary that produces a necessary condition for MIP-representability using recession directions.

Corollary 2.1.5

Suppose S is MIP-representable and $x^* \in Q^n$ is a vector for which there exists some $x^o \in S$ and a sequence $\tau_k \to + \infty$ with

$$x^o + \tau_k x^* \in S \text{ for all } k=1,2, \ldots \qquad (2.1.21)$$

Then there is an $\sigma > 0$, $\sigma \in Q$, with the property that for all $x \in S$ and all integers $p \geq 0$,

$$x + p\sigma x^* \in S, \text{ and} \qquad (2.1.22)$$
$$\sigma x^* \in 2^n(\text{integers}).$$

Rmk: The conditions (2.1.21) are sometimes abbreviated by saying that x^* is a <u>local</u> <u>recession</u> <u>direction</u> of S at x^o.

Proof

By Theorem 2.1.1, for each $k=1,2,\ldots$, there exists $x^k \in S_{a+b}$ and $m^k \in \text{cone}(M)$ with $x^o + \tau_k x^* = x^k + \tau_k m^k$, where $\tau_k m^k \in M$ (so that $m^k \in \text{cone}(M)$). Then note

$$\|x^* - m^k\| < \|x^k - x^0\|/\tau_k < B/\tau_k, \qquad (2.1.23)$$

where B is a bound independent of k.

Since $\tau_k \to +\infty$ we have $x^* \in \text{clcone}(M) = \text{cone}(M)$, as cone(M) is finitely generated [8]. Hence, for some $\sigma > 0$, we have $\sigma x^* \in M$, which implies $\sigma x^* \in Z^n$. Finally, from Theorem 2.1.1, $x + p\sigma x^* \in S$ whenever $x \in S$ and $p > 0$ is integer.

Q.E.D.

Example 2.1.1

Let $S \subseteq Q$ be an MIP-representable set defined by $S = \{x \in Z \mid x = 0 \text{ or } x > 2\}$. An MIP-representation of S is

$x \in S \to$ there exist $u_1, u_2 > 0$, u_1, u_2 integer such that   (2.1.24)
    $x - 2u_1 - 3u_2 = 0$.

with $x^* = 1$, $x^0 = 2$, the hypotheses of Corollary 2.1.5 are met by letting $\tau_k = k$. The conclusion also holds if $\sigma = 2$. It is important to note that we cannot satisfy the corollary merely by making σx^* an integer. E.G., $\sigma = 1$ does not satisfy $x + \sigma x^* p \in S$ for $p=1$ and $x=0$. Thus even integer local recession directions need not be global recession directions (i.e. need not satisfy (2.1.22)).

Suppose we are given representable sets $S_1, \ldots, S_t$. Trivially, their intersection is representable, but what conditions must hold for their union $(S_1 \cup S_2 \cup \ldots \cup S_t)$ to be representable? The next fixed-

charge example illustrates some complexities of this question.

Example 2.1.2

Let f be the fixed-charge function

$$f(x) = \begin{cases} 1 & , \ x > 0 \\ 0 & , \ x = 0. \end{cases} \qquad (2.1.25)$$

Note that epi(f) = $S_1 \cup S_2$ where S_1 and S_2 are polyhedra (thus representable) and defined by

$$S_1 = \{(z,x) | z \geqslant 0, \ x = 0\} \qquad (2.1.26)$$
$$S_2 = \{(z,x) | z \geqslant 1, \ x \geqslant 0\}.$$

However, $S_1 \cup S_2$ is <u>not</u> representable. In fact, with $(z^*, x^*) = (0,1)$ and $(z^o, x^o) = (1,0)$ the hypotheses (2.1.21) of Corollary 2.1.5 hold for S_1 S_2. But the conclusion (2.1.22) fails since $(0,0) \ \varepsilon$ epi(f) and $(0,0) + \gamma(0,1) = (0,\gamma) \notin$ epi(f) for any $\gamma > 0$.

In the above, a recession direction of S_2 (namely $(0,1)$) fails to be a global recession direction of S_1. However, this is not the relevant feature of our example, for the precise relative placement of S_1 with respect to S_2 (and not merely recession directions) also makes a difference. E.G., if $S_1' = \{(-z,1) \ | \ z \geqslant 1\}$ then $(0,1)$ is not a recession direction of S_1' either, but $S_1' \cup S_2$ is MIP-representable (exercise). Because of the relative placement of S_1, S_1' and S_2, we see that $(0,1)$ is

not a global recession direction of $S = S_1 \cup S_2$ (hence S is not representable), yet (0.1) <u>is</u> a recession direction of $S' = S_1' \cup S_2$. This turns out to be the key feature for MIP-representability (see Theorem 2.1.7 below).

The simple construction of the linear system (2.1.27) in our next result will play an important role in our results to follow. This construction derives from disjunctive methods [1], [7].

<u>Lemma 2.1.6</u>

If $S_1, \ldots, S_t$ are bounded MIP-representable sets, then their union is MIP-representable.

<u>Proof</u>

By Corollary 2.1.2, it suffices to show that any union of polytopes is MIP-representable, since each S_i is a union of polytopes.

Let $P_i = \{x | A^i x \geq b^i\}$ be non-empty polytopes for $i=1,\ldots s$. Note that the boundedness property implies that $A^i x \geq 0 \rightarrow x = 0$. A representation of $P_1 \cup \ldots \cup P_s$ is given by

$$x = \sum_{i=1}^{s} x^i; \quad A^i x^i - b^i \lambda_i \geq 0, \quad \lambda_i \geq 0 \quad \text{for } i=1,\ldots,s \quad (2.1.27)$$

$$\sum_{i=1}^{s} \lambda_i = 1 \text{ and } \lambda_i \text{ integer for } i=1,\ldots,s.$$

Q.E.D.

Before presenting our main theorem, we must formally define our use of recession directions. Our definition is restricted to discrete

recession vectors and from this point on, any mention of recession directions implies discrete vectors. (Note this restriction has no effect upon corollary 2.1.5).

An integer vector x^* is a discrete recession direction of a set S if for all $x \in S$ and integers $p > 0$, we have $x + px^* \in S$. The set of all recession vectors is denoted rec(S).

Note that rec(S) is closed under addition, and integer multiples, and $0 \in$ rec(S); thus rec(S) is an integer monoid. [8]

Lemma 2.1.6 is still valid if each polyhedron P_i gives the very same continuous recession directions $\{x \mid A^i x \geq 0\}$, independent of i. In this case, the $P_i \neq \emptyset$ need not be bounded. The same proof of lemma 2.1.6 justifies this claim. In particular, if each function f_i is polyhedral on a nonempty domain $(\text{dom}(f_i) \neq \emptyset)$ and the recession directions of all sets $\text{dom}(f_i)$ are the same for $i \in I(i)$, then the function

$$f(x) = \min\{f_i(x) \mid x \in \text{dom}(f_i)\} \qquad (2.1.28)$$

is MIP-representable.

We next state our main result on when the finite union representable sets is representable.

Theorem 2.17

Suppose that $S_1, \ldots, S_t$ are MIP representable non-empty sets. Then $S = S_1 \cup \ldots \cup S_t$ is MIP-representable if and only if every

discrete recession direction of every S_i has a positive multiple which is a discrete recession direction of S.

Proof

A recession direction x^* of $S_i \neq \emptyset$ satisfies the hypothesis of Corollary 2.1.5. The conclusion of Corollary 2.1.5 must hold if S is MIP-representable, hence our conclusion is necessary.

Next suppose our condition holds: we must show that the corresponding S is MIP-representable. Let the sets $S^i_{a_i+b_i p}$ and M^i and constants a_i, $b_i > 0$ be defined for S_i as in Theorem 2.1.1. From (2.1.4) each integer monoid M_i is finitely generated and therefore M_i has a set of generators $a_{ij} \in R^n$ for $j \in I(i)$, where $I(i)$ is an index set. Each a_{ij} is a discrete recession direction for S_i, and, from our condition, $\sigma_{ij} a_{ij} \in rec(S)$ for some $\sigma_{ij} > 0$. Since $\sigma_{ij} \in Q$, we may take $\sigma_{ij} \in Z$ by clearing denominators. Then $\sigma_i = \prod_{j \in I(i)} \sigma_{ij}$ is an integer and $\sigma_i M_i \subseteq rec(S)$. Furthermore, from Theorem 2.1.1 we have

$$S_i = S^i_{a_i+b_i\sigma_i} + \sigma_i M_i \qquad i=1,\ldots,t \qquad (2.1.29)$$

In fact, we have

$$S = (\bigcup_{i=1}^{t} S^i_{a_i+b_i\sigma_i}) + (\sum_{i=1}^{t} \sigma_i M_i). \qquad (2.1.30)$$

The containment ($\subseteq$) in (2.1.30) is trivial, since if $x \in S$ we have $x \in S_i$ for some $i=1,\ldots,\tau$, and (2.1.29) holds. As to the containment ($\supseteq$) in (2.1.30), let $x = x^0 + x^*$, with $x^0 \in \bigcup_{i=1}^{t} S^i_{a_i + b_i \sigma_i}$ and $x^* \in \sum_{i=1}^{t} \sigma_i M_i$.

Since each $S^i_{a_i + b_i \sigma_i} \subseteq S_i$, we have $x^0 \in S$. Since $\sigma_i M_i \subseteq rec(S)$, we have $x^* \in rec(S)$. Thus $x \in S$.

Each set $S^i_{a_i + b_i \sigma_i}$ is a bounded set, and MIP-representable; thus by Lemma 2.1.6, the union is representable. Since each M_i is finitely generated, so is $\sum_{i=1}^{t} \sigma_i M_i$. By applying the necessary condition of corollary 2.1.4 to (2.1.30), S is MIP-representable.

Q.E.D.

From Theorem 2.1.7, we may determine representability of any finite collection of representable sets using discrete recession directions. Moreover, from the proof of Theorem 2.1.7, it is not necessary that every discrete recession direction of S_i have a positive multiple that is in $rec(S)$; we only need $\sigma_i M_i \subseteq rec(S)$ for a suitable integer $\sigma_i > 1$, where M_i is the integer monoid of (2.1.4). One easily proves that $M_i \subseteq rec(S_i)$, but the converse is false, as our next example shows. This example also illustrates the fact that M_i depends not only on S_i, but also on a specific representation for S_i.

Example 2.1.3

A representation of the set $S = \{x \mid x > 0\}$ is

$$x = x_1 + 2x_2, \quad 0 < x_1 < 2, \quad x_2 > 0, \quad x_2 \in Z. \qquad (2.1.31)$$

Using (2.1.4) to determine M, we find that

$$M = \{x = 2x_2 \mid x_2 > 0, \, x_2 \in Z\}, \qquad (2.1.32)$$

As $x_1 = 0$ is forced in M. We know that a recession direction of S is $x^* = 1$, but in this representation $x^* = 1 \notin M$. (However, whenever our MIP-representation results in a $x^* \in rec(S)$, but $x^* \notin M$, we will always have some $\sigma > 0$, with $\sigma x^* \in M$.) If '2' is replaced by '1' as a coefficient in (2.1.31), the resulting M is indeed all of the recession directions. An alternate form of Theorem 2.1.7 requires the cone of the recession directions of the set S to be equal to the union of the cones of recession directions of each S_i.

Corollary 2.1.8

Let $S_1, \ldots, S_t$ be MIP-representable and put $S = S_1 \cup \cdots \cup S_t$. Then S is MIP-representable if and only if

$$cone(rec(S)) = \bigcup_{i=1}^{t} cone(rec(S_i)) = \sum_{i=1}^{t} cone(rec(S_i)) \qquad (2.1.33)$$

In particular, if S is MIP-representable, the right-hand-side of (2.1.33) is a cone.

Proof:

By Theorem 2.1.7, S is MIP-representable if and only if

$$\text{rec}(S_i) \subseteq \text{cone}\{\text{rec}(S)\} \text{ for } i=1,\ldots,t. \qquad (2.1.34)$$

(Recall that the cone operation is only with respect to rational multipliers).

Suppose S is representable and $x^* \in \text{rec}(S)$. Fix any $x^0 \in S$. Then there is some S_i such that $x^0 + qx^* \in S_i$ for infinitely many $q > 0$. By Corollary 2.1.5, we also have $\sigma x^* \in \text{rec}(S_i)$ for at least one $\sigma > 0$ and $i \in \{1,\ldots, t\}$. Thus,

$$\text{rec}(S) \subseteq \bigcup_{i=1}^{t} \text{cone}\{\text{rec}(S_i)\}. \qquad (2.1.35)$$

Upon taking the cone generated by each set S_i in (2.1.34) and applying (2.1.35), we find

$$\text{rec}(S) \subseteq \bigcup_{i=1}^{t} \text{cone}\{\text{rec}(S_i)\} \subseteq \text{cone}\{\text{rec}(S)\}. \qquad (2.1.36)$$

We now show that $C = \bigcup_{i=1}^{t} \text{cone}\{\text{rec}(S_i)\}$ is a cone. This will also demonstrate that $C = \sum_{i=1}^{t} \text{cone}\{\text{rec}(S_i)\}$.

In fact, if $x, y \in C$ then for certain j, and k, we have $x \in \text{cone}\{\text{rec}(S_j)\}$ and $y \in \text{cone}\{\text{rec}(S_k)\}$. By (2.1.39), $x, y \in \text{cone}\{\text{rec}(S)\}$, so $x + y \in \text{cone}\{\text{rec}(S)\}$; i.e., for some $\sigma \in Q$, we have

$\sigma(x+y) \in rec(S)$. By the first containment in (2.1.36), for some p we have $\sigma(x+y) \in cone\{rec(S_p)\}$ and hence $x+y \in cone\{rec(S_p)\}$. As C is closed under addition (and clearly closed under positive multiples), C is a cone.

By applying the cone operation $rec(S)$ in (2.1.36) we obtain (2.1.33) when S is representable.

Suppose (2.1.33) holds, it implies $cone\{rec(S_i)\} \subseteq cone\{rec(S)\}$ for $i=1,\ldots,t$, in which case (2.1.34) follows, forcing S to be representable.

$$Q.E.D.$$

Note that, as we saw in the fixed-charge example (Example 2.1.2), it is not sufficient for the r.h.s. of (2.1.33) to be a cone. But if the $cone\{rec(S_i)\}$ is independent of $i=1,\ldots,t$, the result holds.

Corollary 2.1.9

If $cone\{rec(S_i)\}$ is independent of $i=1,\ldots,t$, and each S_i is representable, then $S = S_1 \cup \ldots \cup S_t$ is representable.

Proof:

Fix $i=1,\ldots,t$. If $x^* \in cone\{rec(S_i)\}$, then for all $j=1,\ldots,t$ we have $x^* \in cone(rec(S_j))$. Hence, for any j there is some integer $\sigma_j > 0$ for which we have $\sigma_j x^* \in rec(S_j)$.

Define $\sigma = \prod_{i=1} \sigma_i$. Then $\sigma x^* \in rec(S_j)$ for all $j=1,\ldots,t$. Thus $\sigma x^* \in rec(S)$, and the necessary condition of theorem 2.1.7 is verified.

$$Q.E.D.$$

We note, from example (2.1.2), that the sufficiency condition of Corollary 2.1.9 is not necessary.

Proposition 2.1.10

If S is MIP-representable, then S is closed.

Proof:

Suppose $x^n \in S$ for $n=1,2,\ldots$ and $x^n \to x^0$. With $\alpha = \|x^0\| + 1$, S_α is a bounded representable set. By Corollary 2.1.2, S_α is a finite union of polytopes, and thus closed. Since $x^n \in S_\alpha$ for large n, $x^0 \in S_\alpha \subseteq S$.

Q.E.D.

Example 2.1.4

The "fixed-benefit function" f given by

$$f(x) = \begin{cases} 0 & , \quad x = 0 \\ -1 & , \quad x > 0, \quad x \leqslant M \end{cases}$$

for either M finite or infinite, is not MIP-representable, since epi(f) is not closed and Proposition 2.1.10 applies. (Note that $(-1, \frac{1}{n}) \in$ epi(f) for each n but $(1,0) \notin$ epi(f)). (N.B. In the "minimizing format" we have assumed, cost is minimized, so that a benefit or profit shows up as a negative cost. If one switches to a maximizing format, the difficulties will remain).

By the kind of reasoning as in the example, one easily establishes the known result that a representable function is lower semicontinuous.

Example 2.1.5

The "fixed-benefit function with minimum usage level" δ, is given by

$$f(x) = \begin{cases} 0 & , \quad x = 0; \\ -1 & , \quad \delta \leqslant M. \end{cases}$$

For $M > \delta > 0$, and M finite, $f(x)$ is MIP-representable. In fact, one easily shows that epi(f) is the union of two polyhedra which have (1,0) as their sole recession direction, and our earlier remarks apply. In fact, by Theorem 2.2.1 and remarks to follow, f is representable using only binary integer variables in a set-partitioning constraint. (Actually, we can have M infinite and f will be representable; this is a somewhat harder exercise).

We conclude this section with a necessary and sufficient condition for a set S to be MIP-representable.

Theorem 2.1.11

A set S is MIP-representable if and only if S has the form

$$S = (\bigcup_{i=1}^{t} P_i) + M \qquad (2.1.36)$$

for a finite set of polytopes $P_1, \ldots, P_t$ and an integer moncid M with finite basis.

Proof:

Necessity follows from Corollary 2.1.4 and then Corollary 2.12.

Sufficiency follows by Corollary 2.1.4, since $S' = \bigcup\limits_{i=1}^{t} P_i$ is MIP-representable $(rec(P_i) = \{0\}$ as each P_i is a polytope, and e.g. Corollary 2.1.9 can be used).

<div align="right">Q.E.D.</div>

Section 2: Bounded-integer Representability

In this section we explore Meyer's concept of bounded representability. We define S as <u>bounded</u> MIP-representable if there are rational matrices A_1, A_2, A_3 and a vector b, plus a vector bound u^o such that

$$x \in S \leftrightarrow \text{there are } u, v \geqslant 0, u \text{ integer and } u \leqslant u^o, \text{ and} \qquad (2.2.1)$$
$$A_1 x \; A_2 u + A_3 v = b.$$

Note that S need not necessarily be bounded as a set, in order to be bounded MIP-representable; e.g. we can have $S = P_1 \cup P_2$ where P_1 and P_2 are unbounded polyhedra with $rec(P_1) = rec(P_2)$.

Theorem 2.2.1

S is bounded MIP-representable if and only if:

(i)   S is a finite union of polyhedra; and also

(ii) The following condition is satisfied for every vector $x^* \in Q^n$:

If there exists $x^o \in S$ and $\tau_k \to \infty$ with $x^o + \tau_k x^* \in S$ for $k=1,2...,$

then for all $\tau > 0$ ($\tau \in Q$) and all $x \in S$, we have $x + \tau x^* \in S$.

Remark:

The condition in (ii) is, in words, that every local recession direction of S is also a continuous global recession direction of S.

Proof:

First suppose S is bounded MIP-representable. Clearly, S is a union of polyhedra.

When $x^* \neq 0$ is such that there exists an $x^o \in S$ and $\tau_k \to + \infty$ with $x^o + \tau_k x^* \in S$ for all $k=1,2,\ldots$, then there exist u_k, $v_k > 0$ with u_k integer such that

$$A_1(x^o + \tau_k x^*) + A_2 u_k + A_3 v_k = b, \text{ and } u_k \leq u^o \text{ for all } k. \qquad (2.2.2)$$

Dividing both sides of (2.2.2) by $|(\tau_k x^*, u_k, v_k)| \to + \infty$ and using compactness we find that there exists $v^* > 0$ with

$$A_1 x^* + A_3 v^* = 0. \qquad (2.2.3)$$

By rationality of A_1, A_3, x^*, we may assume v^* is rational. By a direct computation, $x + \tau x^* \in S$ whenever $x \in S$ and $\rho > 0$.

For the converse, suppose that S is a finite union of nonempty polyhedra $S = P_1' \cup \ldots \cup P_s'$, and the condition holds. If we utilize the

notation $P_1^{\prime} = \{x \mid A^i x \geqslant b_i\}$, then (2.1.27) provides a bounded representation of S.

In fact, let the set represented by (2.1.27) be denoted $S^{\prime}$. For $i=1,\ldots,s$ by putting $\lambda_i = 1$ and $\lambda_j = 0$ for $j \neq i$, $x^j = 0$ for $j \neq i$, we see that $P_i \subseteq S^{\prime}$. Thus $S \subseteq S^{\prime}$. Next, let $x \in S^{\prime}$, so that (2.1.27) holds for certain x^i and λ_i. For some $k=1,\ldots,s$ we have $\lambda_k = 1$ and $\lambda_i = 0$ for $i \neq k$. Without loss of generality, $k=1$. If $i \neq 1$ and $A^i x^i - b^i \lambda_i \geqslant 0$, then $x^o + \tau x^i \in P_1^{\prime}$ for all $\rho \geqslant 0$ whenever $x^o \in P_i^{\prime}$. By the hypothesis (ii), $\bar{x} + \tau x^i \in S$ whenever $\bar{x} \in S$ and $\tau \geqslant 0$. We have $x^1 \in S$; hence $x^1 + x^2 \in S$; hence $x^1 + x^2 + x^3 \in S$; etc. In this manner, we establish that $x = \Sigma x^i \in S$. Hence $S^{\prime} \subseteq S$. We conclude that $S = S^{\prime}$, as desired.

Q.E.D.

Theorem 2.2.1 excludes non-trivial integer monoids, as well as many other unbounded representable sets, from being bounded MIP-representable.

The proof of the "sufficiency" part of Theorem 2.2.1 reveals that, when S is bounded MIP-representable, it has at least one representation (2.1.27) in which all the integer variables ocurring are binary variables that appear in the same set-partitioning constraint (i.e. $\Sigma_i \lambda_i = 1$).

Section 3: Sharpness of Representations

Another property of MIP-representations involves the "relaxation" of the nonnegative integral variables u of (2.1.1) to nonnegative rational values.

Our interest in this relaxation lies in the fact that, in branch-and-bound algorithms, this relaxation is utilized in the subproblems formed. Obviously, it is desirable for this relaxation to be "as accurate as possible." We next prove that the relaxed representation contains the convex hull of the original set S. This fact places a substantial limitation on the accuracy of the relaxation.

Proposition 2.3.1

Whenever matrices A_1, A_2, A_3, and a vector b exist such that

$$x \in S \rightarrow \text{ there are } u, v \geqslant 0 \text{ with } u \text{ integer and} \quad (2.3.1)$$
$$A_1 x + A_2 u + A_3 v = b,$$

then the following holds:

$$x \in conv(S) \rightarrow \text{ there are } u, v \geqslant 0 \text{ such that} \quad (2.3.2)$$
$$A_1 x + A_2 u + A_3 v = b.$$

Proof:

$x \in conv(S)$ implies there exist $\lambda_i \geqslant 0$ for $i=1,\ldots,n$ such that

$$x = \sum_{i=1}^{n} \lambda_i x_i \text{ where } \sum_{i=1}^{n} \lambda_i = 1 \text{ and} \quad (2.3.3)$$

$$x_i \in S \text{ for } i=1,\dots,n.$$

By taking a summation of the following valid equalities,

$$A_1\lambda_1 x_1 + A_2\lambda_1 u_1 + A_3\lambda_1 v_1 = \lambda_1 b \quad, \quad u_1 \text{ integer}$$

$$\vdots \qquad\qquad\qquad\qquad \vdots$$

$$A_1\lambda_n x_n + A_2\lambda_n u_n + A_3\lambda_n v_n = \lambda_n b \quad, \quad u_n \text{ integer}$$

we get,

$$A_1\left(\sum_{i=1}^{n}\lambda_i x_i\right) + A_2\left(\sum_{i=1}^{n}\lambda_i u_i\right) + A_3\left(\sum_{i=1}^{n}\lambda_i v_i\right) = b$$

which gives (2.3.2).

Q.E.D.

A MIP-representation of S (2.1.1) in which (2.3.2) is bi-conditional (↔) is called "sharp." A sharp representation is as accurate as possible in its linear relaxation. By Proposition (2.1.1), this may not be very accurate; but inappropriate MIP representations can be even less accurate.

Example 2.3.1

The bounded fixed-charge function

$$f(x) = \begin{cases} 0 & , & x = 0 \\ 1 & ; & 0 < x \leq 1 \\ +\infty & ; & x \notin [0,1] \end{cases}$$

has an MIP representation of epi(f) as: "$z \geq \frac{1}{2} \cdot x$, $0 < x \leq 1$, z integer."

The LP relaxation defines the function $\frac{1}{2} \cdot x$ on $[0,1]$, and the epigraph of

$\frac{1}{2} \cdot x$ is larger than conv(epi(f)). Of course, the MIP represntation

"$z \geq x$, $0 \leq x \leq 1$, z integer" is sharp. As we shall see below, MIP

representations in common use are often sharp.

In lemma 2.1.6, we proved that union of polytopes is

representable. We now show that the corresponding representation

depicted by (2.1.27) is sharp.

<u>Proposition 2.3.2</u>

If all the non-empty polyhedra P_i (i=1,...,s) have the same

directions of recession, then (2.1.27) is a sharp representation of

$P_1 \cup \cdots \cup P_s$.

<u>Proof</u>:

We omit the proof that (2.1.27) is a representation of

$P_1 \cup \cdots \cup P_s$. This proof is easily accomplished by, e.g., using our

argument below for the sharpness of the representation, and then noting

that (when the λ_i are all integer) we have $\lambda_i = 1$ below.

To establish sharpness, we need only show one direction of the bi-

conditional, and cite Proposition 2.3.1. (To establish a representation,

we need only show the same direction of the bi-conditional in (2.1.1), as

the converse direction is trivial for (2.1.27). I.e., for the set S

defined in (2.1.27), one trivially has $S \supseteq P_1 \cup \dots \cup P_s$).

In (2.1.27) relax "λ_i integer" to "$\lambda_i \in Q$" for $i=1,\dots,s$.

Then if $\lambda_i > 0$, we have $x_i/\lambda_i = w_i \in P_i$.

We find that $x = \sum_{\lambda_i>0} \lambda_i w^i + \sum_{\lambda_i=0} w^i$, where $w^i \in P_i$ if $\lambda_i > 0$ and w^i

is a recession direction of P_i if $\lambda_i = 0$.

Without loss of generality, $\lambda_1 > 0$ (as $\Sigma\lambda_i = 1$). Then $\lambda_1 w^1 +$

$\sum_{\lambda_i=0} w^i = \lambda_1(w^1 + \sum_{\lambda_i=0} w^i/\lambda_1)$. Since each w^i is a recession direction of

P_1 also, $w' = w^1 + \sum_{\lambda_i=0} w^i/\lambda_1 \in P_1$. Thus, without loss of generality, we

can assume $x = \sum_{\lambda_i>0} \lambda_i w^i$. Hence $x \in \text{conv}(P_1 \cup \dots \cup P_s)$.

Q.E.D.

If the MIP-representation of S' is sharp in Corollary 2.1.4, the

construction gives a sharp representation of S. Thus sharp

representations always exist.

Note that as integer variables are arbitrated (set to zero or to

one) in a branch-and-bound algorithm, the representation (2.1.27) of the

union of bounded sets $S_1 \cup \dots \cup S_t$, as obtained by following the idea of

the previous paragraph, does not correspond usually to subunions.

The juxtaposition of sharp representations for two set S_1 and S_2,

while a representation for $S_1 \cap S_2$, is typically not sharp. E.g. if S_1

is a polyhedron and $S_2 = Z^n$ is the n-dimensional integers, the linear

relaxation of the juxtaposed representations gives the polyhedron S_1 (which is typically not the convex span of the integer points in S_1). Thus the problem, of obtaining sharp representations of intersections of representable sets, subsumes cutting-plane theory for linear integer constraints, and is therefore not expected to have an easy solution except for special cases.

It is sometimes convenient to state the "polyhedral" disjunctive representation (2.1.27) in an alternative form, called the "extreme point" form.

Let $P_i = \text{conv}\{x^{ij} \mid j \in I(i)\} + C$, where the index set $I(i)$ depends on $i = 1, \ldots, s$ and the polyhdedral cone $C = \text{cone}\{v^k \mid k \in K\}$ is independent of i. The "extreme point" representation of $P_1 \cup \ldots \cup P_s$ is:

$$x = \sum_{i=1}^{s} \sum_{j \in I(i)} \lambda_{ij} x^{ij} + \sum_{k \in K} \sigma_k v^k \qquad (2.3.4)$$

$$\lambda_i = \sum_{j \in I(i)} \lambda_{ij}$$

$$1 = \sum_{i=1}^{s} \lambda_i$$

$$\lambda_i \text{ integer}$$

$$\lambda_i, \ \lambda_{ij}, \ \sigma_k \geq 0$$

Proposition 2.3.3

If all the non-empty polyhedra $P_i (i=1,\ldots,s)$ have the same directions of recession, then (2.3.4) is a sharp representation of $P_1 \cup \cdots \cup P_s$.

Proof:

Let S' be the set defined by (2.3.4) in variables $x \in R^n$, and let $S = P_1 \cup \cdots \cup P_s$.

If $x \in P_i$, let λ_{ij}, $\sigma_k \geqslant 0$ be such that $\sum_{j \in I(i)} \lambda_{ij} = 1$ and $x = \sum_{j \in I(i)} \lambda_{ij} x^{ij} + \sum_{k \in K} \sigma_k v^k$. Then by putting $\lambda_i = 1$ and $\lambda_j = 0$ for $j \neq i$, (2.3.4) holds. Hence $S' \supseteq P_i$. As i was arbitrary, $S \subseteq S'$.

If $x \in S'$, suppose that $\lambda_i = 1$ in (2.3.4). Then $\lambda_j = 0$ for $j \neq i$ and so $x = \sum_{j \in I(i)} \lambda_{ij} x^{ij} + \sum_{k \in K} \sigma_k v^k \in P_i$. Thus $x \in S'$, and so $S \supseteq S'$.

We have established that $S = S'$, i.e. that (2.3.4) is a representation of S.

To see that (2.3.4) is exact, only one direction of the biconditional need be established (i.e. that if x solves the linear relaxation, then $x \in conv(S)$).

Put $I = \{i \mid \lambda_i > 0\}$; we have $I \neq \emptyset$. For $i \in I$, define $\lambda'_{ij} = \lambda_{ij}/\lambda_i$. Without loss of generality, $\lambda_1 > 0$. With $q'_k = \sigma_k/\lambda_1$, we have:

$$x = \sum_{i \in I} \sum_{j \in I(i)} \lambda_{ij} x^{ij} + \sum_{k \in K} q_k v^k$$

$$= \sum_{i \in I} \lambda_i \left(\sum_{j \in I(i)} \lambda'_{ij} x^{ij} \right) + \sum_{k \in K} \sigma_k v^k$$

$$= \lambda_1 \left(\sum_{j \in I(1)} \lambda'_{ij} x^{ij} + \sum_{k \in K} \sigma_k v^k \right) + \sum_{\substack{i \in I \\ i \neq 1}} \lambda_i \sum_{j \in I(i)} \lambda'_{ij} x^{ij}$$

We also have , for $i \in I$, $\displaystyle\sum_{j \in I(i)} \lambda'_{ij} = \lambda_i / \lambda_i = 1$.

Upon putting $x^{(1)} = \displaystyle\sum_{j \in I(i)} \lambda'_{ij} x^{ij} + \sum_{k \in K} \sigma_k v^k$, $x^{(i)} = \displaystyle\sum_{j \in I(i)} \lambda'_{ij} x^{ij}$

for $i \in I$ and $i \neq 1$, we easily prove that $x^{(i)} \in P_i \subseteq S$ for $i \in I$. Since

$x = \displaystyle\sum_{i \in I} \lambda_i x^{(i)}$ and $\displaystyle\sum_{i \in I} \lambda_i = 1$, $x \in \text{conv}(S)$, as desired.

Q.E.D.

Note that λ_i can be removed in the linear relaxation of (2.3.4), which becomes

$$x = \sum_{i=1}^{s} \sum_{j \in I(i)} \lambda_{ij} x^{ij} + \sum_{k \in K} \sigma_k v^k \qquad (2.3.5)$$

$$1 = \sum_{i=1}^{s} \sum_{j \in I(i)} \lambda_{ij}$$

$$\lambda_{ij}, \sigma_k \geq 0$$

Only $(n+1)$ constraints appear in (2.3.5). The linear relaxation of (2.1.27) contains many more constraints.

The "common wisdom" regarding the Simplex Algorithm, in practical applications, is that its running time is linear in the number of constraints, and is affected by the number of variables only slightly. Clearly, this practical observation is based on experience where the number of variables is not exponential! For "arbitrary" polyhedra P_i, the number of extreme points x^{ij} (i.e. the size of $I(i)$) $\underline{is}$ exponential, so the exact representation (2.3.4) is of no practical value.

However, a second observation from experience is that, when representations are called for, the P_i which arise are not "arbitrary" at all, and have very few extreme points. Part of the reason for this is that the P_i thus occurring are of small dimension, but that is not the entire reason. In any event, for the common representations there are not many variables λ_{ij} and the $(n+1)$ linear inequalities needed, assumes primary importance. As we shall see in the next section, the most efficient known linear relaxations of common experience - which have often been arrived at without a general method or explicit mention of the sharpness properties of (2.3.5) are instances of (2.3.5).

Section 4: Corollaries and Applications

4.1 Relation to Earlier Results. Section 2 of this chapter contains a definition of bounded-MIP-representable sets that is closely related to the definition used by Meyer [9]. Specifically, a function f is bounded-MIP-representable if and only if f has a rational MIMM with the bounded-integer property of (2.2.1). We will indicate in this section that our Theorem 2.2.1 is a generalization of the five main

theorems used by Meyer for the one-dimensional problem, for the case of rational representability.

The following result is Theorem 2.2.1 of Meyer [9] for the rational case.

Theorem 2.4.1

A function $g(x)$, with a bounded non-empty effective domain contained in R, is bounded MIP-representable if and only if:

> (1) The effective domain is the union of a finite number of closed and bounded intervals;
>
> (2) $g(x)$ is lower semi-continuous;
>
> (3) $g(x)$ is either identically $-\infty$ or finite and piecewise-linear (with a finite number of segments) on its effective domain.

Proof:

A function $g(x)$ with bounded domain satisfies the given conditions if and only if epi(g) is a finite union of polyhedra each with a single, common recession direction, $x^* = (0,1)$. By Theorem 2.2.1, a function with bounded domain is bounded MIP-representable if and only if epi(g) is a finite union of polyhedra with sole recession direction $(0,1)$.

Q.E.D.

Meyer's Theorems 2.3.1 and 2.3.2 concerning semi-infinite domains are proven in a similar manner. However, one and only one polyhedron P_N in the finite union epi(g) $= P_1 \cup \ldots \cup P_N$ will have two extreme recession directions, one of which is $x^* = (0,1)$, and the other unique to the "final" unbounded interval. This additional recession direction

serves to restrict the functions thus representable. For example, the conditions of Theorem 2.3.1 of Meyer involve an effective domain bounded from below. The conditions are:

(1) The effective domain is a closed interval;

(2) $g(x)$ is lower semi-continuous and continuous from the right;

(3) $g(x)$ is either identically $-\infty$ or finite and piecewise-linear on its effective domain;

(4) If $g(x)$ is not identically $-\infty$, then $c_i < c_N$ where c_N is the slope of the "final" (infinite) interval and c_i the slope of of all other intervals.

As before, our Theorem 2.2.1 leads to the condition (3). The new recession direction $(1, c_N)$ of P_N must be a continuous recession direction of $epi(g) = P_1 \cup \ldots \cup P_n$, and the continuous nature of this recession direction will make (1), (2), and (3) necessary. E.g., if g is not lower-semicontinuous, there is $x^0 \in R$ and $\delta > 0$ such that for a sequence $\delta_m \to 0^+$ we have $g(x^0 \pm \delta_m) < g(x^0) - \delta$. Since $(x^0 \mp \delta_m, g(x^0) - \delta) \in epi(f) = P_1 \cup \ldots \cup P_N$ and $epi(g)$ is closed, we have $(x^0, g(x^0) - \delta) \in epi(g)$, which is impossible. For a second instance, if g is not continuous from the right, by lower semi-continuity there exists x^0 and $\delta > 0$ such that for a sequence $\delta_m \to 0^+$, $g(x^0 + \delta_m) > g(x^0) + \delta$. But as $epi(g)$ has a recession direction $(1, c_N)$ we also have $g(x^0 + \delta_m) < g(x^0) + c_N \delta_m$, and thus a contradiction. Our Theorem 2.2.1 can also be directly applied to obtain the sufficiency of (1) - (4).

Meyers Theorem 2.3.2 allows the effective domain to be unbounded from the left. Only condition (4) from above is altered to read:

(4) If $g(x)$ is not identically $-\infty$, then $c_1 \leqslant c_i$ for

$i=2,\ldots,n$, where c_1 is the slope of the "initial (unbounded from below) interval.

Identical arguments from Theorem 2.2.1 also establish this result. Theorem 2.4.1 of [9], involving a domain which is unbounded in both directions, amounts to juxtaposing the two cases of unbounded domains given above.

One advantage of our approach via recession directions, is that it indicates the generalization of these results to functions of more than one variable; and the statement of the result is succinct, rather than involving a list of conditions.

<u>4.2 Application to Other Models</u>. In this section we give an instance in which the inequality description of a function (2.1.27), which results in a disjunctive MIP-representation, becomes Beale's representation of a certain model studied in [2, p. 216] (when an extreme point formulation (2.3.4) of the linear inequality system is employed).

Beale's representation for an approximation of a general function $g(x,y) = xf(y)$, where $f(y)$ is nonlinear, is equivalent to (see [3]):

$$z \geqslant \sum_{i=1}^{2} \sum_{j=1}^{s} \lambda_{ij}\, x_i\, f(\bar{Y}_j) \qquad (2.4.1)$$

$$\sum_{i=1}^{2} \sum_{j=1}^{s} \lambda_{ij} = 1$$

$$x = \sum_{j=1}^{s} (\bar{X}_1 \lambda_{1j} + \bar{X}_2 \lambda_{2j})$$

$$y = \sum_{i=1}^{2} \sum_{j=1}^{s} \lambda_{ij} \bar{Y}_j$$

$$\lambda_{1j} + \lambda_{2j} \leq \lambda_j \quad \text{for } j=1,\ldots,s$$

$$\text{where } \sum_{j=1}^{s} \lambda_j = 1 \text{ and } \lambda_j \text{ binary,}$$

where $\bar{X}_1 = \bar{X}_{min}$ and $\bar{X}_2 = \bar{X}_{max}$ represent the bounds on x and $\bar{Y}_j$ represent the "grid points" for a certain approximation to f(y) over some interval.

Our linear inequality method of representing the same function $g(x,y) = xf(y)$ proceeds in two steps. First, we isolate the following function as a "piecewise-linear" approximation of g:

$$g(x,y) = \quad xf(\bar{Y}_1) \quad \text{if } y = \bar{Y}_1 \text{ and } \bar{X}_{min} \leq x \leq \bar{X}_{max} \qquad (2.4.2)$$

$$\begin{matrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{matrix}$$

$$xf(\bar{Y}_s) \quad \text{if } y = \bar{Y}_s \text{ and } \bar{X}_{min} \leq x \leq \bar{X}_{max}$$

Next note that, for each $j=1,\ldots,s$ the set

$P_j = \{(z,x,y) | z \geq xf(\bar{Y}_j), \bar{X}_{min} \leq x \leq \bar{X}_{max}, y = \bar{Y}_j$ is a polyhedron with

$(1,0)$ as its sole recession direction, and that $epi(\tilde{g}) = P_1 \cup \ldots \cup P_s$. Using (2.1.27) to represent $epi(\tilde{g})$, we obtain:

$$z = \sum_{j=1}^{s} \lambda_j z_j$$

$$x = \sum_{j=1}^{s} \lambda_j x_j$$

$$y = \sum_{j=1}^{s} \lambda_j \bar{Y}_j$$

$$z_j > x_j f(\underline{\bar{Y}}_j)$$

$$\underline{\bar{X}}_{min} \lambda_j < x < \underline{\bar{X}}_{max} \lambda_j$$

$$\sum_{j=1}^{s} \lambda_j = 1 , \lambda_j \text{ integer.}$$

An extreme point representation of (2.4.2) alters expression (2.4.2) as follows. Since the λ_j are binary variables, and $\sum_j \lambda_j = 1$, there are continuous variables λ_{1j} and λ_{2j} with $\lambda_j = \lambda_{1j} + \lambda_{2j}$ and $x_j = \lambda_{1j} \underline{\bar{X}}_{min} + \lambda_{2j} \underline{\bar{X}}_{max}$. We have:

$$x = \sum_{i=1}^{2} \sum_{j=1}^{s} \lambda_{ij} \bar{\underline{X}}_i, \quad \text{where } \bar{\underline{X}}_1 = \underline{X}_{\min}; \; \bar{\underline{X}}_2 = \underline{X}_{\max}; \qquad (2.4.3)$$

$$\lambda_{1j} + \lambda_{2j} < \lambda_j, \; \lambda_j \quad \text{binary and} \quad \sum_{j=1}^{s} \lambda_j = 1;$$

$$\sum_{i=1}^{2} \sum_{j=1}^{s} \lambda_{ij} = 1;$$

$$y = \sum_{j=1}^{s} (\lambda_{1j} + \lambda_{2j}) \; \bar{\underline{Y}}_j$$

$$z > \sum_{j=1}^{s} \sum_{i=1}^{2} \lambda_{ij} \; \bar{\underline{X}}_i \, f(\bar{\underline{Y}}_j)$$

which is identical to (2.4.1). An important note for applications is
that the specific representation used by Beale contains more variables,
but fewer constraints than the linear inequality formulation. Thus for
large problems, Beale's technique is better suited for L.P. pivoting. As
a general rule, we can always use the extreme point representation of the
polyhedra P_i in (2.1.27) in place of the linear inequality
representation, if that is advantageous.

2.4.3. Sharpness in Separable Programming, an Application

In this section we treat the representation of a separable
function $g(x) = \sum_{i=1}^{n} g_i(x_i)$, by approximating each $g_i(x_i)$ by a piecewise-

linear function $\tilde{g}(x_i)$. Clearly, our definition of sharpness for each approximation $\tilde{g}$ (see 2.3.2) and text) is identical to

$$\text{epi}\{\tilde{g}_i^L\} = \text{conv}\{\text{epi}(g_i)\}, \qquad (2.4.4)$$

where $\tilde{g}_i^L$ is the function derived from the LP relaxation of the MIP-representation of $\tilde{g}_i(x)$.

We next show that an approximation of $\tilde{g}_i(x)$, defined on a bounded interval T, may be constructed with a finite union of polytopes, which is MIP-representable. We then adapt the disjunctive MIP-representation used earlier in this thesis (see 2.1.27) and (2.3.4).

Proposition 2.4.2

Let $\tilde{g}(t)$ be a piecewise linear function of one variable t on a closed, bounded interval T. Suppose that $\tilde{g}(t)$ is linear on each interval $[c_{j-1}, c_j]$ (k=1,...,s), where the intervals are disjoint except for end points, and have union T. For each k, define the polyhedron

$$P_k = \{(z,t) \in R^2 \mid t \in [c_{j-1}, c_j], z > \tilde{g}(t)\}. \qquad (2.4.5)$$

Then (2.1.27) and (2.3.4) both define a sharp representation of $\tilde{g}$.

Proof:

This follows from Proposition 2.3.3, since (1,0) is the common and only direction of recession of the polyhedra P_k.

Q.E.D.

We illustrate the function $\tilde{g}$ in Proposition 2.4.2 in fig. 2.1.
Here a_j is the slope of $\tilde{g}(t)$ within the jth interval, $L_j = [c_{j-1}, c_j]$,
and $b_j = \tilde{g}(c_j)$, the end point values of $\tilde{g}(t)$. Using the matrix notation
from lemma 2.1.6, we describe a polyhedral union description of $\tilde{g}(t)$ as:

$$c_{j-1}\lambda_j < t_j < c_j \lambda_j \qquad j=1,\ldots,s \qquad\qquad (2.4.8)$$

$$\lambda_j > 0, \; \sum_{j=1}^{s} \lambda_j = 1; \qquad \text{all } \lambda_j \text{ integers;}$$

$$\sum_{j=1}^{s} t_j = t$$

and enter $z = \sum_{j=1}^{s} (\lambda_j b_{j-1} + a_j t_j)$ in the minimizing criterion function
wherever $\tilde{g}(t)$ is needed.

We next shall adapt the "extreme point" description to our
piecewise-linear example (fig. 2.4.1) to develop an MIP-representation
that also remains "sharp" after binary variables are arbitrated.

The extreme points of P_i in (2.4.5) are (b_{i-1}, c_{i-1}) and (b_i, c_i).
The sole extreme ray $(1,0)$ can be omitted in (2.3.4), since a minimum
value of z is sought. Then the linear relaxation (2.3.5) of (2.3.4)
becomes:
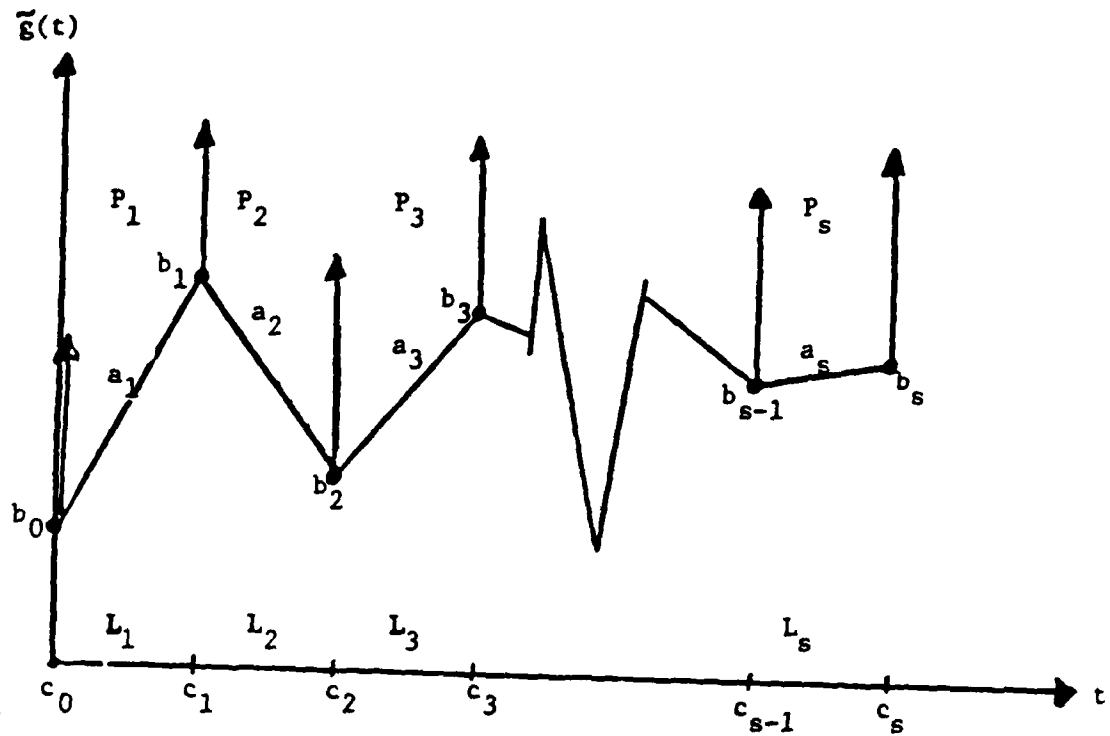
Figure 2.1, Function $\tilde{g}(t)$

$$x = \sum_{i=1}^{s-1} \sum_{j=1}^{2} (\lambda_{i1} c_i + \lambda_{i2} c_{i+1}) \qquad (2.4.9)$$

$$1 = \sum_{i=1}^{s-1} \sum_{j=1}^{2} \lambda_{ij} \quad , \text{ all } \lambda_{ij} \geqslant 0$$

where $z = \sum_{i=1}^{s-1} \sum_{j=1}^{2} (\lambda_{i1} b_i + \lambda_{i2} b_{i+1})$ is entered in the criteron function.

Actually, a small simplification enters in this case, if we set $\Theta_1 = \lambda_{11}$, $\Theta_s = \lambda_{s-1,2}$ and $\Theta_i = \lambda_{i-1,2} + \lambda_{i,1}$ if $1 < i < s$. Then (2.4.9) becomes:

$$x = \sum_{i=1}^{s} \Theta_i c_i$$

$$1 = \sum_{i=1}^{s} \Theta_i \quad , \quad \text{all } \Theta_i \geqslant 0$$

and $z = \sum_{i=1}^{s} \Theta_i b_i$ is entered in the criterion function. This simplification occurs because the right most point of one interval is also the leftmost point of the next interval. If there were a "gap" in the domain, our method would still work, but the simplification would not.

The system (2.4.10) is the most compact relaxation of a formulation that is known for separable programming, and is given in [2].

It involves only two constraints, and a number of variables Θ equal to the number of intervals.

From Proposition 2.3.3, the system (2.4.10) is exactly a formulation for $\tilde{g}^L$. Thus if $\tilde{g}$ is convex (i.e. has convex epigraph), so that $\tilde{g}^L = g$, the system (2.4.10) is a formulation of $\tilde{g}$, as noted in [2]. When $\tilde{g}$ is not convex, [2] recommends a pivot rule which cannot always be used to optimality. Alternatively, one can still solve (2.4.10) and gather what information is possible via this "best possible" linear relaxation.

2.4.4 New Representations. At this time, the best of the existing representations for functions of one variable have nice properties. A primary applications area for our techniques is to functions of several variables.

We focus on a useful function of two variables below, and use rectangular domains. This use of rectangular domains can have limitations, depending on the application, if many variables are to be accommodated (due to the growth in extreme points). However, that difficulty need not arise if simplicial domains can be profitably used. Our example to follow is intended simply to illustrate our approach.

If activity i is employed, a fixed charge f_i is incurred (i=1 or 2). If both are employed, a fixed charge f_b is incurred, and f_b need not be the sum $f_1 + f_2$. Specifically, we are to model the fixed charge function:

$$g(x_1,x_2) = \begin{cases} 0 \, , & \text{if } x_1 = x_2 = 0; \\ f_1, & \text{if } 0 < x_1 \leq M_1, \ x_2 = 0; \\ f_2, & \text{if } x_1 = 0, \ 0 < x_2 \leq M_2; \\ f_b, & \text{if } 0 < x_1 \leq M_1, \ 0 < x_2 \leq M_2; \end{cases} \qquad (2.4.11)$$

The bounds M_1 and M_2 will be necessary, as they were for the one-dimensional fixed-charge problem. As we chose to avoid unbounded integer variables, we seek a bounded integer representation.

It can be shown that, for a representation to exist for (2.4.11), we need these conditions satisfied:

$$0 < f_1, \ 0 < f_2, \qquad \max\{f_1, f_2\} \leq f_b \qquad (2.4.12)$$

In fact, if (2.4.12) fails epi(g) is not closed (see Proposition 1.10). The conditions "$0 < f_i$" simply require that we have a true fixed-charged (rather than fixed-benefit) problem. Also, "$\max\{f_1, f_2\} \leq f_b$" can always be arranged. (If it at first appears that $f_2 > f_b$, always set up for both activities when you set-up for activity two, and use $f_2 = f_b$).

When (2.4.12) holds, we have

$$\text{epi}(g) = P_0 \cup P_1 \cup P_2 \cup P_3 \qquad (2.4.13)$$

where $P_0 = \{(z,0,0) \mid z \geq 0\}$, $P_1 = \{(z,x_1,0) \mid z \geq f_1 \text{ and } 0 < x_1 \leq M_1\}$, $P_2 = \{(z,0,x_2) \mid z \geq f_2, \ 0 < x_2 \leq M_2\}$, and $P_3 = \{(z,x_1,x_2) \mid z \geq f_b, \ 0 < x_1 \leq M_1, \ 0 < x_2 \leq M_2\}$.

We shall investigate the linear relaxation (2.3.5); and as above,

since a minimum value of z is sought we can omit the common recession direction (1,0,0).

We obtain the following for (2.3.5):

$$x_1 = (\lambda_{12} + \lambda_{32} + \lambda_{34}) M_1 \qquad (2.4.14)$$

$$x_2 = (\lambda_{22} + \lambda_{33} + \lambda_{34}) M_2$$

$$\lambda_{01} + \sum_{j=1}^{2} \lambda_{1j} + \sum_{j=1}^{2} \lambda_{2j} + \sum_{j=1}^{4} \lambda_{3j} = 1$$

$$\text{all } \lambda_{ij} > 0$$

where $z = (\lambda_{11} + \lambda_{12})f_1 + (\lambda_{21} + \lambda_{22})f_2 + (\lambda_{31} + \lambda_{32} + \lambda_{33} + \lambda_{34})f_b$ is to be put into the objective function.

To obtain (2.4.14), the following list of extreme points were used, with multipliers $\lambda_{ij} > 0$ as indicated:

$$\lambda_{01} \quad (0,0,0) \qquad P_0$$

$$\left.\begin{array}{ll}\lambda_{11} & (f_1,0,0) \\ \lambda_{12} & (f_1,M_1 0)\end{array}\right\} P_1$$

$$\left.\begin{array}{ll}\lambda_{21} & (f_2,0,0) \\ \lambda_{22} & (f_2,0,M_2)\end{array}\right\} P_2$$

$$\left.\begin{array}{ll}\lambda_{31} & (f_b,0,0) \\ \lambda_{32} & (f_b,M_1,0) \\ \lambda_{33} & (f_b,0,M_2) \\ \lambda_{34} & (f_b,M_1,M_2)\end{array}\right\} P_3$$

There are substantial simplifications in (2.4.14). For example, positive values of λ_{11}, λ_{21}, or λ_{31} deteriorate (raise) the value of x, yet they do not occur in the expressions for x_1 or x_2; and also λ_{01} occurs nowhere except in the last constraint. Thus we may take $\lambda_{11} = \lambda_{21} = \lambda_{31} = 0$. Moreover, if $\lambda_{32} > 0$, by decreasing λ_{32} and increasing λ_{12} amount, we retain feasibility and can only decrease z (since $f_1 < f_b$ by (2.4.12). Thus we may take $\lambda_{32} = 0$. A similar argument allows the simplification $\lambda_{33} = 0$. We have thus eliminated five of the nine variables in (2.4.14), and λ_{01} can eliminated in favor of an inequality convexity constraint. Only three variables remain: λ_{12}, λ_{22}, and λ_{34}. All this depends only on the necessary conditions (2.4.12). We obtain:

$$x_1 = (\lambda_{12} + \lambda_{34})M_1 \qquad (2.4.14)$$

$$x_2 = (\lambda_{22} + \lambda_{34})M_2$$

$$\lambda_{12} + \lambda_{22} + \lambda_{34} \leq 1, \qquad \text{all } \lambda_{ij} \geq 0.$$

where $z = \lambda_{12} f_1 + \lambda_{22} f_2 + \lambda_{34} f_b$ is entered in the objective function.

It can be shown, in the common instance $f_b \leq f_1 + f_2$, that we need not have both λ_{12} and λ_{22} positive when z is minimized. By algebraic substitution, the case $\lambda_{12} = 0$ is equivalent to $x_2/M_2 \geq x_1/M_1$, while the case $\lambda_{22} = 0$ is equivalent to $x_2/M_2 \leq x_1/M_1$. The optimal value $z(x_1,x_2)$ of z in the minimum can be computed to be

$$z(x_1,x_2) = \begin{cases} (f_b-f_2)x_1/M_1 + f_2 x_2/M_2, & \text{if } x_2 M_2 \geq x_1/M_1; \quad (2.4.15) \\ f_1 x_1/M_1 + (f_b - f_1)x_2/M_2, & \text{if } x_2/M_2 \leq x_1/M_1. \end{cases}$$

As we are assuming $f_b \leq f_1 + f_2$ to obtain (2.4.15), we see that always $z(x_1,x_2) \leq f_1 x_1/M_1 + f_2 x_2/M_2$, with equality holding if $f_b = f_1 + f_2$. Note that $f_1 x_1/M_1 + f_2 x_2/M_2$ is the sum of the individual linear relaxations when joint effects are ignored. In the case $f_1 = f_2 = f_b = f$ that one set-up pays for both activities, (2.4.15) gives

$$z(x_1,x_2) = f(\max\{x_1/M_1, \ x_2/M_2\}) \qquad (2.4.16)$$

which is a formula that can be used directly to replace the system (4.14). (This also can be done with (2.4.15.)

In the less usual case $f_b > f_1 + f_2$ that a special "additional penalty" is levied when both activities are undertaken, we note that

$$z(x_1,x_2) = f_1 x_1/M_1 + f_2 x_2/M_2 \text{ whenever} \qquad (2.4.17)$$
$$x_1/M_1 + x_2/M_2 \leq 1,$$
$$x_1, x_2 \geq 0$$

Indeed, $\lambda_{12} = x_1/M_1$, $\lambda_{22} = x_2/M_2$, $\lambda_{34} = 0$ solves (2.4.14) and gives this value, while any increase in λ_{34} produces a larger value of z (consider λ_{34} as a non-basic variable, and note that its reduced cost is $f_b - f_1 - f_2 > 0$). If $x_1/M_1 + x_2/M_2 > 1$, we must have $\lambda_{34} > 0$ in (2.4.14). Then from $f_b > f_1 + f_2$, one can show that $\lambda_{12} + \lambda_{22} + \lambda_{34} = 1$. Upon solving the resulting three linear equations for z we obtain:

$$z(x_1,x_2) = (f_1 + f_2 - f_b) + (f_b - f_2)x_1/M_1 + (f_b - f_1)x_2/M_2 \qquad (2.4.18)$$
$$\text{whenever } x_1/M_1 + x_2/M_2 > 1 \; ; \; x_1, x_2, \geq 0$$

For the expression (2.4.18), it can be shown that $z(x_1,x_2) > f_1 x_1/M_1 + f_2 x_2/M_2$ in the region described, i.e. the relaxation value is higher than the sum of the independent values (as it should be).

The algebraic simplifications performed mechanically above only amount to removing extreme points of a P_i which is no longer extreme in $K = \text{conv} (P_0 \cup P_1 \cup P_2 \cup P_3)$. In an easily visualized formulation (2.4.11) like the present, geometric intuition can speed the process.

Since K is a polyhedron, the linear relaxation $z(x_1,x_2)$ of g is the maximum of a finite set of linear forms. Here, more than two forms

was not needed. The expressions for $z(x_1, x_2)$ obtained in this way can be used to replace the system (2.4.14), if that is desired. The linear forms involve are exhibited above; we leave the details to the reader, and note only that this process, too, is part of a general procedure.

REFERENCES

1. Balas, E., "Disjunctive Programming," in Discrete Optimization II, eds. P. L. Hammer, E. L. Johnson, B. H. Korte, North-Holland Publishing Co., Amsterdam, pp. 3-53, 1979.

2. Bazaraa, M. and Shetty, M., Non-Linear Programming, John Wiley & Sons, Inc., New York 1979.

3. Beale, E. M. L., "Branch-and-Bound Methods for Mathematical Programming," in Discrete Optimiztion II, eds. P. L. Hammer, E. L. Johnson, B. H. Korte, North-Holland Publishing Co., Amsterdam, pp. 201-221, 1979.

4. Dantzig, G., Linear Programming and Extensions, Princeton University Press, 1963.

5. Garfinkel, R. S. and G. L. Nemhauser, Integer Programming, John Wiley & Sons, Inc., New York, 1972.

6. Jeroslow, R. G., "Cutting Plane Theory: Algebraic Methods," Discrete Mathematics, Vol, 23, pp. 121-150, 1978.

7. Jeroslow, R. G., "Cutting Plane Theory: Disjunctive Methods," Annals of Discrete Mathematics, Vol. 1, pp. 293-330, 1977.

8. Jeroslow, R. G., "Some Basis Theorems for Integral Monoids," Mathematics of Operations Research, Vol. 3, No. 2, pp. 145-154, 1978.

9. Meyer, R. R., "Mixed Integer Minimization Models for Piecewise Linear functions of a Single Variable," Discrete Mathematics, Vol. 16, pp. 163-171, 1976.

10. Meyer, R. R., "Integer and Mixed-Integer Programming Models: General Properties," Journal of Optimization Theory and Applications, Vol. 16, Nos. 3/4, pp. 191-206, 1975.

11. Meyer, R. R., "On the Existence of Optimal Solutions to Integer and Mixed Integer Programming Problems," Mathematical Programming, Vol. 7, pp. 223-235, 1974.

12. Rockafeller, R. T., Convex Analysis, Princeton University Press, 1970.

13. Stoer, J., and Witzgall, C., Convexity and Optimization in Finite Dimensions I, Springer-Verlag, New York, 1970.

CHAPTER III

EXPERIMENTAL RESULTS ON THE NEW TECHNIQUES
FOR INTEGER PROGRAMMING FORMULATIONS

Section 1: Introduction

A widely accepted view in integer programming research is that
progress, in utilizing (mixed-) integer formulations to solve real-world
problems, will rely primarily on algorithm advances (for general and
special structures) and clever coding tricks. In this view, we are
perceived as already knowing how to represent a real-world problem with
integer variables; those simple "formulation techniques" appear early in
the subject and by now are widespread in the master's level and even
undergraduate level textbooks. (see e.g. [7]) However, since in the
1970's, the experience of practitioners indicates that some major issues
of formulation have been overlooked (see e.g. [8], [17], [23], [24],
[25]).

In this chapter, we are going to provide furthur experimental
results, which favor certain new integer formulations that we introduced
in chapter II. Our formulations are not ad hoc "practitioners tricks,"
but derive from a systematic study of modelling. These formulations are
either directly derived from, or motivated by, developments in the
"disjunctive methods," particularly as in [2] and [14] (see [13] for a
survey of these methods). R. R. Meyer [19], [20], [21], [22] and

T. Ibaraki [12] initiated the systematic study of "integer modellings";
see also [15].

We caution that our experimental results are preliminary: the
problems studied are "small" by industrial standards. Nevertheless, the
advantage of the newer formulations does appear to increase with problem
size. Also the initial linear program solved is typically much tighter
than in standard formulations, a fact which is code-independent. We use
the Land and Powell code because of its accessibility and excellent
documentation. We also run problems using C. H. Martin's BANDBX code,
and CDC's APEX IV mixed-integer code.

We do not contend that the newer formulations will necessarily be
better, in terms of CPU time, than those previous. They often are not,
for example, on tiny problems (fewer than 10 binary variables).
Moreover, of the potentially limitless number of different formulations
from the real world, we study only a few common ones. Nevertheless, the
new modelling techniques are quite "automatic," easy to learn and to
apply (either by hand, or in a "wrap-around" of a standard MIP code), and
seem to markedly improve performance in some common problems.

The modelling techniques we have developed have these properties:
(1) They are automatic, and do not depend on an ad hoc analysis of the
problem to be modelled; (2) The linear relaxation is optimal for the set
modelled; (3) Under mild assumptions, variable arbitration in branch-and-
bound leads to a modelling of our type, so that reformulations are not
needed in lower nodes of the search tree.

For related work on problem formulation, see [4], [8], [17], [23], [24], [25].

Here is the plan of this chapter.

In Section 2, we quickly review a few basic concepts and results from chapter II, and also introduce the idea of "modelling linkage." We explain the "sharpness" concept, which is part of what differentiates our modellings from those previous. This is sufficient for the reader to go on to the description of our experiments, and to our experimental results. In Appendix A, we provide (what is essentially) a tutorial on those results and techniques from chapter II which are relevant here. The appendix is optional reading.

Section 3 presents an experiment on sharpness, which we test in the setting of a multi-divisional firm (like that considered by Dantzig and Wolfe [6]) where, moreover, each division has a choice of technologies at its disposal. This variant of "multiple choice constraints" is tried, since some hand calculations with these kinds of constraints indicate a dramatic lack of sharpness, in the formulations given in most articles and common textbooks. Our experiment confirms a dramatic advantage for the newer formulations.

Section 4 presents an experiment on modelling linkage, which we test in the setting of separable concave functions with multiple fixed-charges. Our experimentation here is more limited, but it confirms a definite, but smaller, advantage of the newer modellings.

Section 3.2: What the Experiments Test

We begin this section with a short summary of some theoretical results.

A set S of rationals in Q^n is called <u>bounded-MIP-representable</u>, if there are rational matrices A_1, A_2, A_3 and a rational vector b, as well as a bound B, such that:

$$x \in S \rightarrow \text{ there are } u, v \geqslant 0, \text{ with } u \text{ integer} \qquad (3.2.1)$$
$$\text{and } ||u|| \leqslant B, \text{ such that}$$
$$A_1 x + A_2 u + A_3 v = b$$

Functions f which occur only in a minimization objective are represented via their epigraph $epi(f) = \{(z,x) \mid z \geqslant f(x)\}$; their domains may be a general subset of Q^n. This will then coincide with Meyer's definition of function representability [18] in rationals. (Caution: Functions occuring otherwise may require representation by graphs or hypergraphs).

A more general concept of MIP representability is given in chapter II but we do not use it here. Moreover, unlike some previous work (e.g. [19]), all our representations are in the rational field, and are of sets of rationals.

Here is a general result chapter II, which characterizes bounded-MIP-representability.

<u>Theorem</u>: The set $S \subseteq Q^n$ is bounded-MIP-representable, if and only if, both these conditions hold:

α) S is a finite union of polyhedra;

β) Whenever $x^o \in S$ and $y \in Q^n$ are such that

x^o + ny ϵ S for all n=1,2,3, ..., then for

every x ϵ S and all $\lambda > 0$ we have x + λy ϵ S.

By the Theorem, any finite union of (rational) polytopes (bounded polyhedra) is bounded-MIP-representable, since always y=0 is forced when the hypothesis of β) holds, so that β) is vacuous. The Theorem also shows, that the unbounded fixed-charge problem is not bounded-MIP-representable. (Actually, Meyer earlier showed it is not representable at all). The function for this charge is

$$f(x) = \begin{cases} 0 & , \quad x = 0; \\ \\ c & , \quad x > 0, \end{cases} \qquad (3.2.2)$$

with c > 0. We have epi(f) = $P_1 \cup P_2$, with P_1 = {(z,x) | x=0, z $\geq$ 0}, P_2 = {(z,x) | x $\geq$ 0, z $\geq$ c}, so that condition α) of the Theorem holds. However, with (z^o, x^o) = (c,0), y^o = (0,1), and (z,x) = (0,0), we find that condition β) fails. This explains why we need an upper bound on x to get a representation for fixed-charges.

The Theorem "almost" says, but does not actually say, that when S is written as a union of polyhedra S = $P_1 \cup P_2 \cup ... \cup P_t$, then all P_i have the same recession directions. For example, if t=2, P_1 = {(x_1, x_2) | 0 $\leq$ x_1 $\leq$ 1, 0 $\leq$ x_2 $\leq$ 1}, P_2 = {(x_1,x_2) | x_1 $\geq$ 1, 0 $\leq$ x_2 $\leq$ 2,}, then by the Theorem S = $P_1 \cup P_2$ is representable, even though P_1 has no recession directions and P_2 has (1,0) as a recession direction. (The exact "placement" of P_1 relative to P_2 matters; for example, if

$P_1 = \{(x_1, x_2) \mid -1 \leqslant x_1 \leqslant 0, \, 0 \leqslant x_2 \leqslant 1\}$, then $S = P_1 \cup P_2$ is not bounded-MIP-representable). However, if C is the sum of the recession cones of the P_i for a bounded-MIP representable set, note that $S = (P_1 + C) \ldots (P_t + C)$. Hence, "without loss of generality," the $P_i' = P_i + C$ can be taken to have the same recession directions.

Note that some important sets are not bounded-MIP-representable (indeed, not representable at all), as for example the "complementarity set":

$f(x_1, x_2) = \{(x_1, x_2) \geqslant 0 \mid x_1 \cdot x_2 = 0\} = P_1 \cup P_2$, where

$P_1 = \{(x_1, x_2) \geqslant 0 \mid x_1 = 0\}$ and $P_2 = \{(x_1, x_2) \geqslant 0 \mid x_2 = 0\}$.

(One easily verifies that β) fails). Meyer and Thakker have introduced other representability concepts for such a set [22], the concept of "polyhedral union" representability, which will not concern us here.

Our focus in this chapter is not on the issue of the existence of representations, but on certain qualities they may have, when they exist. The first significant quality is that of "sharpness."

A specific bounded modelling A_1, A_2, A_3, b for a set S is called sharp, if, in addition to (3.2.1):

$$x \in \text{clconv}(S) - \text{there are } u, \, v \geqslant 0, \qquad (3.2.3)$$
$$\|u\| \leqslant B, \text{ with}$$
$$A_1 x + A_2 u + A_3 v = b.$$

In (3.2.3), the integrality condition is dropped ("relaxed") on u, and

clconv(S) is the closure of the convex span of S. In chapter II, it is shown (and it is an easy result) that, whenever (3.2.1) holds, then

$$x \in clconv(S) \rightarrow \text{there are } u, v > 0, \ ||u|| < B, \quad (3.2.4)$$

with

$$A_1 x + A_2 u + A_3 v = b.$$

Therefore, a modelling is exactly sharp if its linear relaxation is as "tight" (i.e. small) as possible.

In branch-and-bound codes, the algorithm uses the linear relaxation of a mixed integer program, both for fathoming and for guiding the search strategy. Therefore one might heuristically conclude that sharp formulations should be superior to nonsharp ones. This is not an exact deduction, since sharp modellings may contain a different (often, but not always, larger) number of variables and constraints. Moreover, so many heuristic devices are used in branch-and-bound, which are not (in some sense) "monotone" with the size of the linear relaxation, that an exact analysis seems very difficult. However, the heuristic principle is clear enough. Moreover, we discover that many of the "textbook formulations" are not sharp, so an experiment seems in order.

The concept of "sharpness" is introduced by Meyer [21] for functions, with the larger descriptor, "linear relaxation optimal." In chapter II, we provide sharp formulations for all bounded-MIP-representable (really, all MIP-representable) sets, which also retain sharpness in many (but not all) situations as the variables are

arbitrated by a branch-and-bound code. The second property, which one might call "hereditary sharpness," is important when one cannot intervene in the code at each node, to "touch up" the representation. Our formulations are presented, with worked examples, in Appendix A.

We now wish to introduce the newer concept, of "modelling linkage."

Representations are not used by themselves, but as parts of larger programs. For example, we may have a program:

$$\text{min:} \quad cx \qquad\qquad (3.2.5)$$

$$\text{subject to:} \quad Ax = b, \; x \geqslant 0$$

$$x_j \text{ integer, } j \epsilon J$$

$$x \epsilon S_1$$

$$\text{or}$$

$$x \epsilon S_2$$

When both S_1 and S_2 are bounded-MIP-representable, and in fact

$$x \epsilon S_i \quad \text{—} \quad \text{there are } u^i, \; v^i \geqslant 0 \text{ with } u^i \text{integer,} \qquad (3.2.6)$$

$$||u^i|| \leqslant m_i, \text{ and}$$

$$A_1^i x + A_2^i u^i + A_3^i v^i = b^i$$

then we can solve (2.5) via this mixed-integer program:

$$\text{min:} \quad cx \qquad\qquad (3.2.7)$$

$$\text{subject to:} \quad Ax = b$$

$$A_1^1 x + A_2^1 u^1 + A_3^1 v^1 = b^1$$
$$A_1^2 x + A_2^2 u^2 + A_3^2 v^2 = b^2$$
$$x, u^1, u^2, v^1, v^2 \geq 0$$
$$u^1, u^2 \text{ integer}$$
$$x_j \text{ integer, } j \in J.$$

Note that the constraints of (3.2.7) actually give a bounded-MIP-represention of the set

$$S = \{x \geq 0 \mid Ax = b, \quad x \in S_1 \cup S_2, \quad x_j \text{ integer for } j \in J\}.$$

However, it is important to realize that this may _fail_ _to_ _be_ a _sharp_ presentation of S, _even_ _if_ one begins with sharp representations of S_1 and S_2. In particular, the representation of S obtained by our methods (which give sharp representations) may, and often do, differ from the constraints of (3.2.7).

Modelling linkage refers to the way that the representations of smaller side conditions (like $x \in S_j$) are made independently, and then simply "attached" or "linked into" the whole program (3.2.5). By sharpness considerations, one ideally would like to represent all of a program at once (the set S in the example above), rather than by "pieces" which are attached on. However, generally the size of "large chunk" formulations can grow multiplicatively, so some compromise is often needed. (However, see later for an instance where the "large chunk" formulation is smaller.)

The issue of modelling linkage can be a very subtle one, as, for example, a representation of $S_1' = \{x \in S \mid x_j \text{ integer for } j \in J\}$ can be

superior to one of S_1 alone. It is best clarified in a more general setting (similar to that of the co-propositions of [14]). Modelling linkage is quite clearly related to the issue of sharpness, but it goes further, in that it concerns the <u>setting</u> in which sharpness is sought.

In addition to the "sharpness" aspect of modelling linkage, there is a second subtlety subsumed in this concept, which we call "variable co-ordintion." For example, if a piecewise-linear function of one variable, which also has fixed-charges, is to be modelled in two separate sections -one for the piecewise-linear part, and one for the fixed charges - the separate modelling may not take explicit care to insure that the same interval of function value is being considered in one part as in the other.

Particularly, after some integer variables controlling segments of the function have been arbitrated to specific binary values, it is quite possible that in some subproblems of the branch-and-bound tree, different segments are being considered in the separate modellings. This fact need not show up as a simple inconsistency of the linear program for that subproblem, but it can greatly multiply (unnecessarily) the size of the branch-and-bound tree. We say in such cases that the segment variables are "unco-ordinated," and such a lack of co-ordination would not occur in a simultaneous modelling of both aspects of the function together.

Incidentally, in some cases variable co-ordination can be achieved among the separate parts of the modelling, without going to our techniques; but this phenomenon of co-ordination seems not to have been discussed previously. Variable co-ordination refers primarily to the

hereditary properties of modellings (i.e. how they behave as integer variables are arbitrated).

For a preliminary test on modelling linkage, and hence also variable co-ordination, we choose a problem in separable programming of the type discussed above. Each objective function involves a single variable, and is piecewise-linear, except for "fixed-charges" (or, in the economist's terminology, "set-up charges" - as all our costs are variable prior to the decision made in connection with the model).

We use four formulations: the first two are "separate" formulations, one for the fixed charges and one for the piecewise-linear segments , which are separately attached to the main program. They are similar to formulations found in textbooks. The third formulation is our sharp formulation for the entire function (done at one modelling). The fourth formulation is of the "separate" variety, and as each separate part comes from our techniques, each separate part is both sharp and hereditarily sharp. (One of the textbook-like formulations is not sharp, although it did quite well; the second is not hereditarily sharp.)

As it turns out, the third formulation, which involves the entire function, also requires half as many binary variables - an example of how a smaller program can sometimes have a tighter linear relaxation! By the usual rule of thumb, the third formulation ought to run faster, and its improved sharpness should increase the effect. However, we run the experiment to be certain, and it does confirm our expectations.

Incidentally, the data of the type used in our second experiment is such, that we believe our favorable results are due to the variable

co-ordination aspect of the simultaneous modelling, and not its sharpness property.

Section 3.3: The Experiment for Multiple Choice Constraints

We begin this section by describing a set of problems that will allow us to empirically compare, our modelling for either/or constraints with the standard textbook modelling. In our formulation, these mixed integer programs have binary integer variables in special ordered sets, where for each set of binary variables at most one is non-zero. However, we do not incorporate special-ordered set branching.

The scenario for the test problems involves a corporation which has several divisions producing different end products. Each division has the choice of different technologies in producing these products, and the products produced by individual divisions are different. The technology chosen affects the total output and product mix of each division. Since the corporation must satisfy financial and capacity restriction considerations which apply to all products produced, its goal is to simultaneously select the desired technology for each division while optimizing corporate profits.

The following notation is used in the problem:

ND   =   number of Divisions.

$NT(i)$ =   number of Technologies possible for Division i.

$NP(i)$ =   number of Products produced by Division i.

$NC(i,j)$ = number of constraints in Division i when using Technology j.

x_i = vector of products produced in Division i, of dimension NP(i)

c_i = contribution of Division i's products to the overall corporate profit (a vector with dimension equal to x_i)

A_i = a matrix of dimension (CCx NP(i)) which is the effect of Division i's output in constraints.

CC = number of common constraints which tie all Divisions together.

D_{ij} = A matrix of dimension (NC(i,j) x NP(i)) which represents the technology of Division i when Technology j is selected.

r_{ij} = The right-hand-side constraining Division i, when Technology j is chosen.

All matrices A_i, D_{ij} are nonnegative and with no zero columns. The general problem is depicted below:

(MIP)   max: $c_1^T x_1 + c_2^T x_2 + \ldots + c_{ND}^T x_{ND}$

ST:

(Common
Constraint) $A_1 x_1 + A_2 x_2 + \ldots + A_{ND} x_{ND} \leq b$

(Division 1)

$$D_{1,1} x_1 \leq r_{1,1} \quad \text{or} \quad D_{1,2} x_1 \leq r_{1,2} \quad \text{or} \ldots$$

$$\ldots \quad D_{1,NT(1)} x_1 \leq r_{1,NT(1)}$$

(Division 2)

$$D_{2,1}x_2 \leqslant r_{2,1} \quad \text{or} \quad D_{2,2}x_2 \leqslant r_{2,2} \quad \text{or} \quad \ldots$$
$$\ldots \quad D_{2,NT(2)}x_2 \leqslant r_{2,NT(2)}$$

$$.$$
$$.$$
$$.$$

(Division ND)

$$D_{ND,1}x_{ND} \leqslant r_{ND,1} \quad \text{or} \quad D_{ND,2}x_{ND} \leqslant r_{ND,2} \quad \text{or} \quad \ldots$$

$$D_{ND,NT(ND)}x_{ND} \leqslant r_{ND,NT(ND)}$$

$$x_i \geqslant 0 \text{ for all } i=1, ND$$

We use two MIP Representations to model this problem. The first is a standard procedure found in numerous sources, such as Eppen-Gould [7]. The second representation is from chapter II on modelling a finite union of polyhedra.

Assuming the maximum production capacity of each division is finite results in each constraint set being a bounded polyhedron, or polytope. In order to model the problem using the standard method, the upper bound vector x_i^* for variable x_i must be known, so that one can calculate upper bounds u_{ij} for each constraint (this procedure is explained in detail later on). For now we define u_{ij} as the value of the r.h.s. of each constraint of Division i, and Technology j, when all products of Division i are producing at this peak value (x_i^*)

$$D_{i,j}x_i^* = u_{i,j}.$$

Thus the standard representation (MIPS) is as follows.

(MIPS) $\qquad$ max: $\underset{\sim}{c}_1^T \underset{\sim}{x}_1 + \underset{\sim}{c}_2^T \underset{\sim}{x}_2 + \ldots + \underset{\sim}{c}_{ND}^T \underset{\sim}{x}_{ND}$

ST:

(Common) $A_1 \underset{\sim}{x}_1 + A_2 \underset{\sim}{x}_2 + \ldots + A_{ND} \underset{\sim}{x}_{ND} \leqslant \underset{\sim}{b}$

(Division 1)

$$D_{1,1} \underset{\sim}{x}_1 + (\underset{\sim}{u}_{1,1} - \underset{\sim}{r}_{1,1}) z_{1,1} \leqslant \underset{\sim}{u}_{1,1}$$

$$D_{1,2} \underset{\sim}{x}_1 + (\underset{\sim}{u}_{1,2} - \underset{\sim}{r}_{1,2}) z_{1,2} \leqslant \underset{\sim}{u}_{1,2}$$

$$\vdots$$

$$D_{1,NT(1)} \underset{\sim}{x}_1 + (\underset{\sim}{u}_{1,NT(1)} - \underset{\sim}{r}_{1,NT(1)}) z_{1,NT(1)} \leqslant \underset{\sim}{u}_{1,NT(1)}$$

(S.O.S Constraint) $z_{1,1} + z_{1,2} + \ldots + z_{1,NT(1)} = 1$

$$\vdots$$

(Division ND)

$$D_{ND,1} \underset{\sim}{x}_{ND} + (\underset{\sim}{u}_{ND,1} - \underset{\sim}{r}_{ND,1}) z_{ND,1} \leqslant \underset{\sim}{u}_{ND,1}$$

$$D_{ND,2} \underset{\sim}{x}_{ND} + (\underset{\sim}{u}_{ND,2} - \underset{\sim}{r}_{ND,2}) z_{ND,2} \leqslant \underset{\sim}{u}_{ND,2}$$

$$\vdots$$

$$D_{ND,NT(ND)} \underset{\sim}{x}_{ND} + (\underset{\sim}{u}_{ND,NT(ND)} - \underset{\sim}{r}_{ND,NT(ND)}) z_{ND,NT(ND)} \leqslant \underset{\sim}{u}_{ND,NT(ND)}$$

(S.O.S. Constraint) $z_{ND,1} + z_{ND,2} + \ldots + z_{ND,NT(ND)} = 1$

(Binary) $\qquad$ $z_{i,j} \in \{0,1\}$ for all i,j

For the MIPS representation, the constraint and variable totals
are: MIPS Constraints $= CC + \sum\limits_{i=1}^{ND} \sum\limits_{j=1}^{NT(i)} NC(i,j) + ND$,

MIPS Variables $= \sum\limits_{i=1}^{ND} \{NP(i) + NT(i)\}$,

$$\text{MIPS Binary} \quad = \quad \sum_{i=1}^{ND} NT(i).$$

The second representation is designed specifically for representing the finite union of a set of polyhedra via integer variables, and is our sharp representation from chapter II. For our division/technology problem, the polyhedra are actually bounded polytopes, and the bounded-MIP representation becomes as follows.

(MIPP) max: $c_1^T x_1 + c_2^T x_2 + \ldots + c_{ND}^T x_{ND}$

ST:

(Common)

$A_1 x_1 + A_2 x_2 + \ldots + A_{ND} x_{ND} \leqslant b$

$x_{1,1} + x_{1,2} + \ldots + x_{1,NT(1)} = x_1$

$\cdot$
$\cdot$
$\cdot$

$x_{2,1} + x_{2,2} + \ldots + x_{2,NT(2)} = x_2$

$x_{ND,1} + x_{ND,2} + \ldots + x_{ND,NT(ND)} = x_{ND}$

(Division 1)

$D_{1,1} x_{1,1} - r_{1,1} z_{1,1} \leqslant 0$

$\cdot$
$\cdot$
$\cdot$

$D_{1,NT(1)} x_{1,NT(1)} - r_{1,NT(1)} z_{1,NT(1)} \leqslant 0$

(S.O.S. Constraint)

$z_{1,1} + z_{1,2} + \ldots + z_{1,NT(1)} = 1$

(Division 2)

$D_{2,1} x_{2,1} - r_{2,1} z_{2,1} \leqslant 0$

$\cdot$
$\cdot$
$\cdot$

$D_{2,NT(2)} x_{2,NT(2)} - r_{2,NT(2)} z_{2,NT(2)} \leqslant 0$

(S.O.S. Constraint)

$z_{2,1} + z_{2,2} + \ldots + z_{2,NT(2)} = 1$

$\cdot$
$\cdot$
$\cdot$

$D_{ND,1} x_{ND,1} - r_{ND,1} z_{ND,1} \leqslant 0$

Division ND)

$$\vdots$$

$$D_{ND,NT(ND)} x_{ND,NT(ND)} - r_{ND,NT(ND)} z_{ND,NT(ND)} \leq 0$$

(S.O.S. Constraint) $z_{ND,1} + z_{ND,2} + \cdots + z_{ND,NT(ND)} = 1$

(Binary Constraint) $z_{i,j} \in \{0,1\}$

Notice that the total variables and constraints of the (MIPP)
representation are, or can be, greater than for (MIPS). However, the
number of binary variables is the same in both formulations. Here are
some counts:

$$\text{(MIPP) Total Constraints} = CC + \left\{ \sum_{i=1}^{ND} \left(\sum_{j=1}^{NT(i)} NC(i,j) \right) + NP(i) \right\} + ND$$

$$\text{(MIPP) Total Constraints} = \text{(MIPS) Total Constraints} + \sum_{i=1}^{ND} NP(i)$$

$$\text{(MIPP) Total Variables} = \left\{ \sum_{i=1}^{ND} \left(\sum_{j=1}^{NT(i)} NP(i) \right) + NT(i) + NP(i) \right\}$$

$$\text{(MIPP) Total Variables} = \text{(MIPS) Total Variables} + \sum_{i=1}^{ND} \sum_{j=1}^{NT(i)} NP(i)$$

$$\text{(MIPP) Binary Variables} = \text{(MIPS) Binary Variables} = \sum_{i=1}^{ND} NT(i).$$

The upper bound vectors x_j^* used to determine the u_j values for the
standard representation (MIPS), are data-dependent on the constraint
coefficients $(D_{i,j})$. Rather than explain their calculation, a small
example best decribes this computation. Suppose Division 1 has the
following situation,

$$\left. \begin{array}{l} d_{1,1} x_1 + d_{1,2} x_2 \leq r_1 \\ \\ d_{2,1} x_1 + d_{2,2} x_2 \leq r_2 \end{array} \right\} \quad \text{or} \quad \left\{ \begin{array}{l} d'_{1,1} x_1 + d'_{1,2} x_2 \leq r'_1 \\ \\ d'_{2,1} x_1 + d'_{2,2} x_2 \leq r'_2 \end{array} \right.$$

Note that $d_{k,L}$ is an element of $D_{1,1}$ and $d'_{k,\ell}$ is similarily an element of $D_{1,2}$ of MIP. Likewise $[r_1, r_2]^T$, and $[r'_1, r'_2]^T$ respectively are $\underset{\sim}{r}_{1,1}$ and $\underset{\sim}{r}_{1,2}$ respectively of MIP.

To determine x_1^* and x_2^* we compare the following ratios;

$$x_1^* = \max\{\min(r_1/d_{1,1} \ r_2/d_{2,1}) \ ; \ \min(r'_1/d'_{1,1} \ ; \ r'_2/d'_{2,1})\}$$

$$x_2^* = \max\{\min(r_1/d_{1,2} \ ; \ r_2/d_{2,2}) \ ; \ \min(r'_1/d'_{1,2} \ ; r'_2/d'_{2,2})\}$$

with all $d_{i,j}$ and $d'_{i,j}$ non-zero. If a d_{ij} or d'_{ij} is zero, the corresponding terms are omitted (recall that all matrices are nonnegative).

We set each x_i^* to its least upper integer bound (to keep within integer data) and then find $\underset{\sim}{u}_{i,j}$ as follows.

$$\underset{\sim}{u}_{i,j} = D_{i,j} \, \underset{\sim}{x}_i^{**}, \text{ where } \underset{\sim}{x}_i^{**} = \lceil \underset{\sim}{x}_i^* \rceil$$

In our example,

$$\underset{\sim}{u}_{1,1} = \begin{bmatrix} u_{1,1} \\ u_{1,2} \end{bmatrix} \begin{array}{l} = (d_{1,1})x_1^{**} + (d_{1,2})x_2^{**} \\ = (d_{2,1})x_1^{**} + (d_{2,2})x_2^{**} \end{array}$$

$$\underset{\sim}{u}_{1,2} = \begin{bmatrix} u'_{1,1} \\ u'_{1,2} \end{bmatrix} \begin{array}{l} = (d'_{1,1})x_1^{**} + (d'_{1,2})x_2^{**} \\ = (d'_{2,1})x_1^{**} + (d'_{2,2})x_2^{**} \end{array}$$

A sample problem with 3 divisions and 2 Technologies/Division, 3 products/Division, 3 constraints/Technology/Division, and 3 common constraints is included in Appendix B.

The idea behind our approach to the setting of upper bounds, is that the quantities x_1^*, x_2^*, etc. are the best valid upper bounds, if we know only that one of the alternative divisional constraints hold (but we don't know which one). While the bounds for $u_{1,1}$, $u_{1,2}$, etc., may seem extravagant, actually they are approached in the worst case.

Therefore, our procedure represents a reasonably "tight" bounding method, which the user could quickly implement. The user might obtain even better bounds by running many LP's, etc., but our method is fast and does <u>not</u> use the large bounds which are encouraged by some authors for the standard formulation.

It has been observed in practice, that excessively large bounds make the standard formulation virtually useless in algorithms. Thus our experimental results concern a <u>different</u> phenomenon, since we provide reasonable bounds for the standard formulation. Moreover, the total failure of sharpness in the usual formulation will occur even if optimal upper bounds are used.

We look at <u>three</u> <u>criteria</u> in evaluating the results of our computations. We monitor the <u>solution</u> <u>time</u> of each sample, the total <u>number</u> <u>of</u> <u>nodes</u> of the branch-and-bound tree solves as LP's, and the <u>ratio</u> of LP relaxation optimum to the actual mixed integer optimum.

The solution time of each test problem is recorded in CPU seconds. CPU time involves only the actual computer time used in solving the

problem. The time recorded excludes input and output times since these are negligible on large problems.

The number of nodes solved as LPs includes the actual number of LP's solved, including the initial LP relaxation. The larger the number, the further "down" the tree, or the larger the branch-and-bound tree becomes before an optimal solution is either be found, or verified.

Finally the LP solution to optimal MIP solution ratio measures the accuracy of the initial LP relaxation to the actual MIP solution. This ratio may be very essential for problems in which only an approximate solution is necessary, or for codes which use different branching rules, variable choice rules, etc. than does ours.

Our test problems range from a 3 Division, 2 Technologies per Division, 3 common constraints, 3 constraints per Technology, and 3 products per Division; to a 15 Division, 3 Technologies per Division problem with all other parameters the same. While relatively small, these problems give significantly different results for all three test criteria between (MIPS) and (MIPP). The largest representation contains 153 constraints, 210 variables with 45 discrete (binary) variables, of which 30 are declared binary. The number of binary variables actually used in our tests is less than indicated in (MIPS) and (MIPP). This is accomplished by designating one variable of each Special Ordered Set constraint as continuous. Clearly this does not affect the optimal solution, but will reduce the number of binary variables.

The cost coefficients in the objective function, and all constraint coefficients (for all A and D matrices) are randomly generated

integers with bounds [0,10]. The r.h.s. vectors for the divisional constraints ($r_{i,j}$) are calculated as,

$$r_{i,j} = 2*(\text{sum of row coefficients}) + 1.$$

The vector b of the common constraints is set to $\alpha *$(sum of row coefficients), and we use values $\alpha = 1.1$, 1.5, and 1.9 in our study. For the value $\alpha = 1.9$, the common constraints are virtually not binding, so that the time comparisons become essentially a comparison of the modelling techniques.

RESULTS. The following tables outline the results of our test problems. Table 1 contains information concerning problems in which the common constraint α value is $\alpha = 1.1$. Similarily Tables 2, and 3 respectively have α values of 1.3 and 1.9, respectively. Problems in Table 4 are run using Martin's BANDBX code. For the Problem column, the following notation is employed: $n_1 - n_2$ means that there were Divisions, with n_2 Technologies per Division. All problems have 3 products/Division, 3 common constraints, 3 constraints/Technology/ Division. The column headed (#) gives the number of problems in the sample for this size; and the column (#1) indicates the number in the sample which are solved by the first linear program.

The results support our expectation that the (MIPP) representation requires fewer branch-and-bound subproblems (nodes) before obtaining an optimal solution than the (MIPS) representation. At $\alpha = 1.1$, the average node advantage of the (MIPP) representation ranges from 2 to 1 for the smallest sized problems (3-2), to over 13 to 1 for problems (8-3). We run two large problems (12-3) and (15-3) a limited number of times (only

Table 1. Multi-Division Problems
R.H.S. Multiplier (1.1)

Problem	#	Time		Nodes						Ratio LP/Discrete	
				MIPP			MIPS				
		MIPP	MIPS	Avg.	max	#1	Avg.	max.	#1	MIPP	MIPS
3-2	12	.625	0.39	1.833	3	5	3.75	7	1	1.0005	1.084
5-2	12	1.86	2.36	2.5	6	2	9.917	19	0	1.0047	1.098
8-2	12	3.579	11.518	2.166	5	2	29.08	65	0	1.0007	1.174
3-3	12	1.638	0.913	3.916	12	3	6.67	13	1	1.0048	1.088
5-3	12	3.154	6.933	3.75	7	2	24.25	57	0	1.00275	1.102
8-3	12	8.847	33.5	5.16	10	1	69.25	165	0	1.0008	1.1336
12-3	9	21.431	484*	4.11	9	0	140*	-	-	1.00168	1.1466**
15-3	4	29.1	399*	5	6	0	382*	-	-	1.00046	1.144**

*Only one sample
**The ratio is the LP over the Best Solution found

Table 2. Multi-Division Problems
R.H.S. Multiplier (1.3)

Problem	#	Time		Nodes						Ratio LP/Discrete	
				MIPP			MIPS				
		MIPP	MIPS	Avg.	max	#1	Avg.	max.	#1	MIPP	MIPS
3-2	18	0.83	0.69	2.2	4	6	4.9	7	0	1.0066	1.142
5-2	18	1.92	2.72	2.3	4	3	9.0	29	0	1.0009	1.118
8-2	18	5.89	18.60	2.7	7	3	39.6	91	0	1.0013	1.174
3-3	18	1.69	1.52	3.4	10	3	9.7	17	0	1.0032	1.133
5-3	18	4.20	11.52	3.9	10	1	38.2	118	0	1.0011	1.174
8-3	18	10.99	123.5	4.5	18	4	194.7	597	0	1.0017	1.116
12-3	3	19.4	-	6.3	9	0	-	-	-	1.00008	-
15-3	3	30.5	-	4	9	1	-	-	-	1.00008	-

Table 3. Multi-Division Problems
R.H.S. Multiplier (1.9).

Problem	#	Time		Nodes						Ratio LP/Discrete	
				MIPP			MIPS				
		MIPP	MIPS	Avg.	max	#1	Avg.	max.	#1	MIPP	MIPS
3-2	8	0.97	1.44	2.0	3	2	8.75	14	0	1.0026	1.186
5-2	8	2.32	10.42	2.0	3	1	32.75	55	0	1.0006	1.2383
8-2	8	6.60	64.92	3.0	6	3	115.4	278	0	1.0034	1.1926
3-3	8	1.59	4.13	2.38	3	0	21.1	31	0	1.0088	1.1971
5-3	8	5.57	41.77	4.12	9	2	107.7	337	0	1.0041	1.2393
8-3	8	12.22	311.68	3.63	7	1	407.4	762	0	1.0013	1.2283
12-3	3	25.0	325*	3.66	4	0	218*	-	-	1.0004	1.1408
15-3	4	35.15	306*	7	14	0	270*	-	-	1.0008	1.315**

*only one sample
**The ratio is the LP over the best discrete found when stopped.

one sample of each using the (MIPS) representation) to support our

premise that the node advantage increases with problem size. We actually

stop the computer before the (MIPS) model is solved to avoid excessive

computer charges.

At $\alpha = 1.9$ the nodal comparison shows greater advantages for our

(MIPP) representation, ranging from 4:1 for problem (3-2) to over 100-1

on problem (8-3). Again, the single (MIPS) models of size (12-3) and

(15-3) are run for 5 minutes of CPU run time before we stop the code.

Recall that for $\alpha = 1.9$ the common constraints have almost no effect at

all, and many problems solve as LP's in the newer (MIPP) formulation.

We are surprised at the dominance of the (MIPP) model with

respect to run time. Only the smallest problems are run faster using the

(MIPS) representation. At $\alpha = 1.1$, the time advantage reaches 4-1 for

problem (8-3) and continues to increase for the two larger samples (12-

3), (15-3).

Again at $\alpha = 1.3$ and 1.9 the time advantage is even more

significant, reaching 12 to 1, and 25 to 1 respectively for problem (8-

3).

Table 4 summarizes problems solved using the BANDBX code. The

problems are all of size 10 Divisions, 3 Products per Division, 3

Technologies per Division, 3 constraints per Technology, and 3 common

constraints. The α value is 1.3 for all problems in Table 4. For the

five problems run using the BANDBX code, our (MIPP) formulation finds the

discrete solution much faster (in CPU seconds) and more efficiently with

respect to branch-and-bound node counts. The standard (MIPS)

Table 4. BANDBX Code with RHS Multiplier (1.3)
10 Divisions, 3 Products per Division.

Problem (10 Divisions)	TIME (CPU seconds)				Number of Nodes				LP/DISCRETE Ratio	
	MIPP		MIPS		MIPP		MIPS			
	LP	Total	LP	Total	To Find	Total	To Find	Total	MIPp	MIPS
1	36.95	161.8	2.29	300*	8	9	222	325*	1.006	1.34**
2	46.37	54.6	1.84	258.1	2	2	14	261	1.0008	1.12
3	44.10	120.4	3.13	300*	5	5	240	360*	1.003	1.27**
4	41.4	41.4	2.29	300*	1	1	71	298*	1	1.28**
5	40.0	40.0	2.6	300*	1	1	52	324*	1	1.28**

*did not solve the problem before computer shutoff
**ratio is the LP over the Best Discrete Solution found
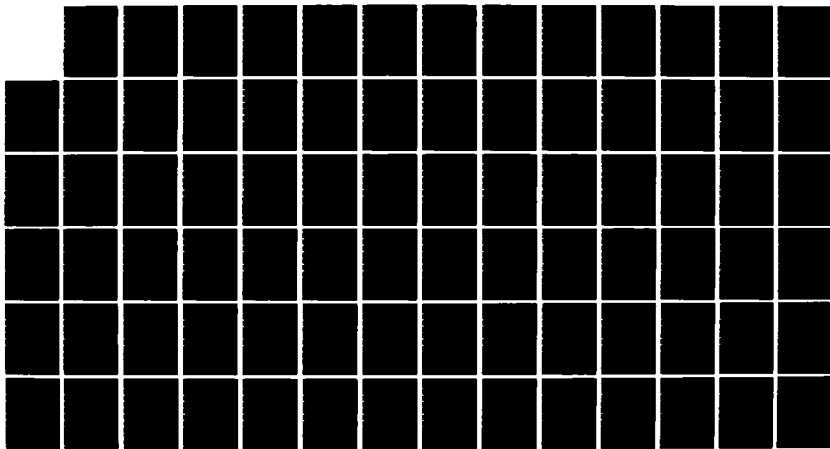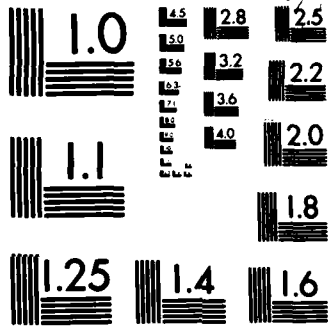
MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

formulation, as Table 4 shows, solves only one of the five samples within our 5 minute CPU time limit. Incidently, we continue to run problem #1 for 10 CPU minutes without verifying an integer optimum. Because of the time limit imposed, it is not possible to compare _exact_ solution time or node counts between the two formulations. Even so, comparing the number of branch-and-bound nodes solved favors our (MIPP) formulation by at least 30-1 for the worst case sample. This worst case result occurs in sample #1 in which the (MIPS) LP/DISCRETE ratio is very high (1.34). The significance of the (LP/DISCRETE) ratio is also evident in that our (MIPP) formulation results in an LP relaxation solution within 1 percent of the optimal discrete solution. On the other hand, the best (LP/DISCRETE) ratio of the (MIPS) formulation is 1.12, which represents a 12 percent error. Incidently, the only problem solved using the (MIPS) formulation also has the lowest (LP/DISCRETE) ratio. With the advantage increasing with problem size, tremendous time savings seem possible using our formulation for even small-sized industrial problems. We caution, however, that the problems in our study are randomly generated, and do not derive from an actual industrial application.

Of course, a more efficient computer code would solve our small problems much faster, probably with fewer nodes. But the third test statistic, the LP relaxation solution-to-mixed integer optimal solution ratio, is _invariant_ with respect to individual computer codes, heuristics used, and whether or not a problem is solved to proven optimality.

For all α values, our (MIPP) representation has an initial LP solution within 1% of the final mixed-integer solution value. On the

other hand, the (MIPS) approach errors by at least 8% on the average, and the error appears to increase with problem size.

Section 3.4: The Experiment for Piecewise-Linear Separable Programs

with Fixed-Charges

The scenario for these test problems involves a manufacturer which must meet the demand of several separate categories of products. For each category, the company assembles up to three end product categories, each of which meets the specification of that category, and only that category. Any combination of end products in a certain category may be used to meet the demand requirement of that category.

For example, a computer may be designed to meet certain specifications. Several configurations of off-the-shelf intermediate products may represent different end products, all of which meet the same specifications. In that case, all the end products fall into the same category.

Each end product is assembled from a specific combination of intermediate products. These intermediate products are common to all end products. The intermediate products are high demand items each with its own cost function. The goal of the company is to meet demand of each category while minimizing its cost of intermediate products. We do not include assembly costs, i.e. we take the perspective of the intermediate-products' manufacturer.

The cost function of each intermediate product is a separable linear function with three separate fixed-charges. For instance, the cost function of the ith intermediate product is:

$$
f_i(x_i) = \begin{cases}
0 & \text{if } x_i = 0; \quad (3.4.1) \\[2ex]
f_{0i} + s_{1i}{}^{*}x_i; & \text{if } x_i \in (0, B_{1i}]; \\[2ex]
f_{0i} + s_{1i}{}^{*}B_{1i} + f_{1i} + (x_i - B_{1i}){}^{*} s_{2i} & \text{if } x_i \in (B_{1i}, B_{2i}]; \\[2ex]
f_{0i} + s_{1i}{}^{*} B_{1i} + f_{1i} + (B_{2i} - B_{1i}){}^{*} s_{2i} + \\[1ex]
f_{2i} + s_{3i}{}^{*} (x_i - B_{2i}) & \text{if } x_i \in (B_{2i}, B_{3i}];
\end{cases}
$$

We consider that the fixed charges are incurred when some technology reaches (physical) "capacity," and a second technology must be used. We graph the function of (3.4.1) as Figure 3.1.
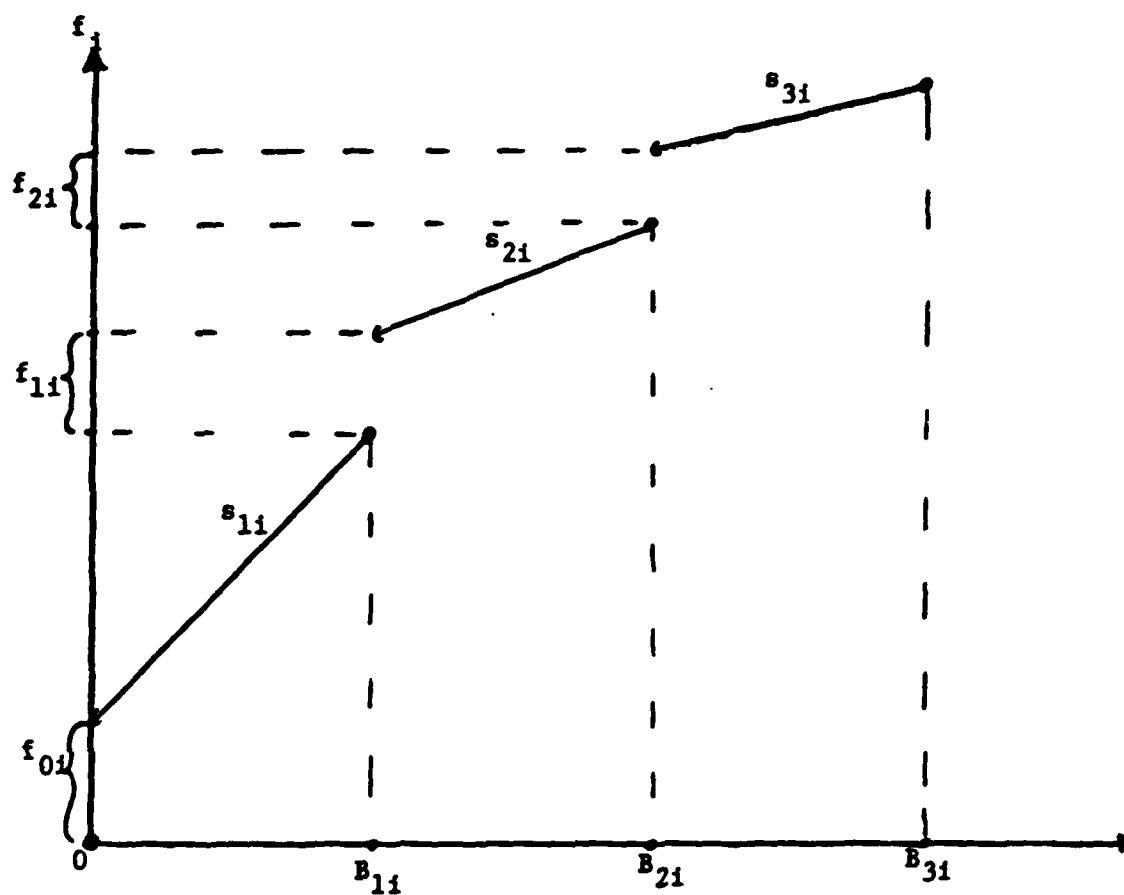
Figure 3.1, Function f_i

Notation for the problem is included below:

S_k = set of similar end products meeting the specifications of category k. $|S_k| < 3$.

NC = Total number of categories

d_k = demand of category k.

y_j = The jth end product made from a specific combination of intermediate products.

NYt = The total number of end products.

x_i = The ith intermediate product

$f_i(x_i)$ = The cost function of the ith intermediate product. Represented by Eqn (3.4.1) and Figure (3.1).

NXi = The total number of intermediate products.

$n(i,j)$ = Matrix representation of the combination of x_i required in the assembly of each y_j.

The problem is now modelled as an optimization problem:

Fixed-Charge Problem

$$\min: \sum_{i=1}^{NXi} f_i(x_i) \qquad (3.4.2)$$

s.t.:

(Product Mix) (I.) $\sum_{j=1}^{NYt} n(i,j)*y_j = x_i$ for i=1, NXi

(Demand) (II.) $\sum_{j \in S_k} y_j > d_k$ for k=1, NC

all vars. > 0

The variables in (3.4.2) are not required to be integral, as it is assumed that the values of x_i and y_{ij} are sufficiently large to allow such a continuous "approximation."

A standard method of representing a fixed charge problem involves removing the fixed charges and separately representing the remaining function. The economies of scale of each intermediate product results in the remaining function being a separable linear function (concave), i.e. $s_{1i} > s2_i > s_3$ for all i=1, NXi. After the remaining separable function is represented, the fixed charges are represented as a separate MIP representation. Finally the fixed charge representation is "added-on" to the separable representation. Thus the original fixed charge problem is the combination of two individual MIP-representations, the models have been "Linked" together.

We use a common standard procedure of representing the separable function, which we draw in Figure 3.2.
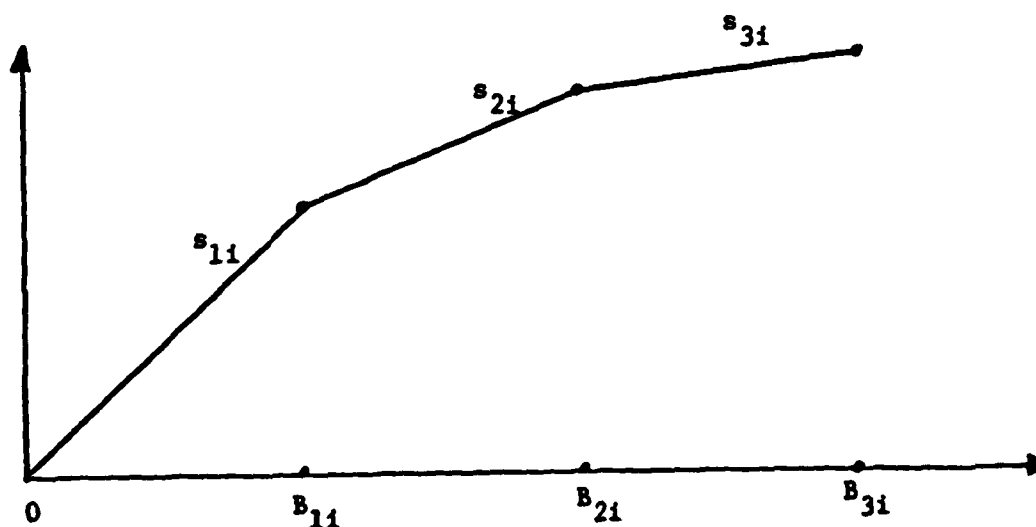
Figure 3.2, Separable Function

To model the piecewise-linear function of Figure 3.2, we insert into the objective function this term:

$$\sum_{i=1}^{NXi} \{\lambda 1_i * s_{1i} * B_{1i} +$$

(3.4.3.a)

$$\lambda 2_i * (s_{1i} * B_{1i} + s_{2i} * (B_{2i} - B_{1i}) +$$

$$\lambda 3_i * (s_{1i} * B_{1i} + s_{2i} * (B_{2i} - B_{1i}) + s_{3i} * (B_{3i} - B_{2i}) \}$$

The following are inserted in the constraints:

$$x_i - (\lambda 1_i * B_{1i} + \lambda 2_i * B_{2i} + \lambda 3_i * B_{3i}) = 0 \quad \text{for } i=1, NXi \quad (3.4.3.b)$$

$$\lambda 0_i + \lambda 1_i + \lambda 2_i + \lambda 3_i = 1$$

$$\lambda 0_i + \lambda 1_i \qquad - \Theta 1_i \qquad \geqslant 0 \qquad \text{for } i=1, NXi$$

$$\lambda 1_i + \lambda 2_i \quad - \Theta 2_i \qquad \geqslant 0$$

$$\Theta 1_i + \Theta 2_i + \Theta 3_i = 1$$

$$\Theta 1_i, \ \Theta 2_i, \ \Theta 3_i \geqslant 0 \text{ integer}$$

This modelling (3.4.3) is not hereditarily sharp since e.g., $\Theta 3_i = 0$ does not imply $\lambda 3_i = 0$. It is simply one which is similar to those in common use.

We represent the fixed-charge step function using two standard techniques, (I) and (II). By separately "adding" (I) and (II) representations to the representation in (3.4.3) above, we have two commonly used MIP-representations of our original problem (3.4.2).

For "Fixed Charge I," we put into the objective function the term:

$$\sum_{i=1}^{NXi} \{f_{0i}*z0_i + f_{1i}*z1_i + f_{2i}*z2_i\} \tag{3.4.4.a}$$

and we put into the constraints:

$$x_i - B_{3i}*z0_i \leq 0 \tag{3.4.4.b}$$

$$x_i - (B_{3i} - B_{1i})*z1_i \leq B_{1i} \qquad \text{for } i=1, NXi$$

$$x_i - (B_{3i} - B_{2i})*z2_i \leq B_{2i}$$

$$z0_i, \ z1_i, \ z2_i \geq 0 \text{ integer}$$

When (Fix-Charge I) is "added" to (3.4.3) we have our first standard MIP representation, we call the representation (STANDARD I), or just (I). Incidentally, (3.4.4) is not sharp for general bounds and fixed-charges. The second technique commonly used to represent a step function adds an additional variable for each fixed-charge. This representation (3.4.5) is sharp and is similar to the "δ method" contained in Bradley, Hax, Magnanti [5]. For "Fixed-Charge 2," we put into the objective the term:

$$\sum_{i=1}^{NXi} \{f_{0i}*z0_i + f_{1i}*z1_i + f_{2i}*z2_i\} \tag{3.4.5.a}$$

and we add in the constraints:

$$x_i - (x1_i + x2_i + x3_i) = 0 \qquad\qquad (3.4.5.b)$$

$$x1_i - B_{1i}*z0_i \qquad\qquad < 0$$

$$x2_i - (B_{2i} - B_{1i})*z1_i \qquad < 0 \qquad\qquad \text{for } i=1, NXi$$

$$x3_i - (B_{3i} - B_{2i})*z2_i \qquad < 0$$

$$x1_i - B_{1i}*z1_i \qquad\qquad > 0$$

$$x2_i - (B_{2i} - B_{1i})*z2_i \qquad > 0$$

$$z0_i, \; z1_i, \; z2_i \qquad\qquad > 0 \text{ integer}$$

By adding (Fix-Charge II) to the (3.4.3), we have the second common MIP-representation of the original problem, which we call this (STANDARD II), or (II) for short.

Notice that the (II) representation has the same number of integer variables as (I), but more total constraints and continuous variables. Therefore we expect that, unless the branch-and-bound node count of (II) is much less than the count for (I), problems will be solved faster using the first method (I). In fact, this turns out to be the case, showing that variable counts can sometimes overcome a deficiency of sharpness.

Note that in both (I) and (II), the fixed-charge variables zj_i are separable from the segment variables Θj_i, which results (as we see below) in an unnecessary doubling of the number of binary variables. However, we cannot validly set, e.g. $z0_i = \Theta1_i$, for possibly $\lambda1_i = 0 = z0_i$, $\Theta1_i = \lambda0_i = 1$. This problem can be overcome by an ad hoc device in this specific setting, but such devices go beyond what one would do in using the standard modellings as they have been described.

We next use the extreme point form of our sharp representation to model the entire function of Figure 3.1. Using our sharp representation there is no need to separate the fixed-charges as before, and we actually require fewer integer variables and fewer total constraints and variables than either of the common methods.

We put into the objective function:

$$\sum_{i=1}^{NXi} \{\lambda 1_i * f_{0i} + \lambda 2_i * (f_{0i} + f_{1i} + s_{1i} * B_{1i}) +$$

$$\lambda 3_i * (f_{0i} + f_{1i} + f_{2i} + s_{1i} * B_{1i} + s_{2i} * (B_{2i} - B_{1i}) +$$

$$\lambda 12_i * s_{1i} * B_{1i} + \lambda 22_i * (s_{2i}) * (B_{2i} - B_{1i}) +$$

$$\lambda 32_i * s_{3i} * (B_{3i} - B_{2i})\}$$

(3.4.6.a)

and add to the constraints:

(III.) $\quad (\lambda 12_i + \lambda 21_i) * B_{1i} + (\lambda 22_i + \lambda 31_i) * B_{2i}$

$\quad\quad + \lambda 32_i * B_{3i} = x_i \quad$ for $i=1$, NXi

(3.4.6.b)

(IV.)

$\quad \lambda 1_i - (\lambda 11_i + \lambda 12_i) = 0$

$\quad \lambda 2_i - (\lambda 21_i + \lambda 22_i) = 0 \quad\quad$ for $i=1$,NXi

$\quad \lambda 3_i - (\lambda 31_i + \lambda 32_i) = 0$

$\quad \lambda 1_i + \lambda 2_i + \lambda 3_i \quad\quad < 1$

$\quad \lambda 1_i, \lambda 2_i, \lambda 3_i > 0$ integer

This formulation is worked in detail in Appendix A.

We conclude with a fourth modelling also involving the two separate parts of the function of Fig. 3.1. This modelling differs from Standards I and II, in that both parts are separately done by our "extreme point representation" (as described in Appendix A), so each part separately exhibits sharpness and hereditary sharpness. This fourth modelling is called "SEMI-SHARP" (for lack of a better name!), and it is in the comparison between SHARP and SEMI-SHARP that we isolate the issue of modelling linkage. The other comparisons provide information which, while involving linkage, also are compounded with other effects.

SEMI-SHARP uses this formulation for the piecewise-linear part of the function:

$$\sum_{i=1}^{NXi} \{\lambda12_i + \lambda2_i + \lambda3_i s_{1i} * B_{1i} \qquad (3.4.7.a)$$

$$+ (\lambda22_i + \lambda3_i) * s_{2i} * (B_{2i} - B_{1i})$$

$$+ \lambda32_i * s_{3i} * (B_{3i} - B_{2i})\}$$

is added into the objective function, while these relations are added as constraints for $i = 1, \ldots, NXi$:

$$x_i = (\lambda12_i + \lambda21_i)B_{1i} + (\lambda22_i + \lambda31_i)B_{2i} + \lambda32_i B_{3i} \qquad (3.4.7.b)$$

$$\lambda11_i + \lambda12_i = \lambda1_i$$

$$\lambda21_i + \lambda22_i = \lambda2_i$$

$$\lambda31_i \ \lambda32_i = \lambda3_i$$

$$\lambda 1_i + \lambda 2_i + \lambda 3_i = 1$$

$$\lambda 1_i, \ \lambda 2_i, \ \lambda 3_i \text{ are integer}$$

Also, all variables in (3.4.7.b) are nonnegative.

SEMI-SHARP uses this formulation for the fixed-cost part of the function:

$$\sum_{i=1}^{NXi} \{\Theta 1_i f_{0i} + \Theta 2_i (f_{0i} + f_{1i}) \qquad (3.4.8.a)$$

$$+ \Theta 3_i (f_{0i} + f_{1i} + f_{2i})\}$$

is added into the objective function, while these relations are added as constraints for all i:

$$x_i = (\Theta 12_i + \Theta 21_i)B_{1i} + (\Theta 22_i + \Theta 31_i)B_{2i} + \Theta 32_i B_{3i} \qquad (3.4.8.b)$$

$$\Theta 11_i + \Theta 12_i = \Theta 1_i$$

$$\Theta 21_i + \Theta 22_i = \Theta 2_i$$

$$\Theta 31_i + \Theta 32_i = \Theta 3_i$$

$$\Theta 1_i + \Theta 2_i + \Theta 3_i \leqslant 1$$

$$\Theta 1_i, \ \Theta 2_i, \ \Theta 3_i \text{ integer}$$

Also, all variables in (3.4.8.b) are nonnegative.

For this problem, the sharp representation requires one-half of the integer variables than those required by either of the standard representations.

Table 5. Problem Sizes of Models.

Model	Total #Constraints	# Variables	Integer Variables
STANDARD I	10 * NXi + NC	13 * NXi	6 * NXi
II	13 * NXi + NC	16 * NXi	6 * NXi
SHARP	6 * NXi + NC	12 * NXi	3 * NXi
SEMI-SHARP	11 * NXi + NC	19 * NXi	6 * NXi

While the primary goal of these experiments involves "modelling linkage" comparisons, we also compare a heuristic commonly used in applications of mixed integer programming. This heuristic attempts to reduce the number of branch-and-bound nodes solved in verifying a feasible mixed integer solution within 2% of the optimal solution. The heuristic fathoms a branch-and-bound limb if the current problem solution is either infeasible, integer-feasible, or within 2% of the best known mixed integer solution.

Several problems are tested using three different mixed integer programming codes; Land-Powell, C. H. Martin's BANDBX, and CDC's APEX IV.

Test Problems

The test problems generated have the following parameter characteristics:

$NXi = 5, 6, 10$

$NYt = 2*NXi$

$NC = [NYt/3]$

$d_k \epsilon [100, 200]$ integer, random

$n(i,j) \epsilon [0, 5]$ integer, random

$s_{1i} \epsilon [1,5]$ real, random

$s_{2i} = RES*s_{1i}$

$s_{3i} = RES*s_{2i}$

$RES = 0.8$

$S_1 = \{1,2,3\}$

$S_2 = \{4,5,6\}$

.

.

.

$$S_k = \{\text{remaining end products}\}$$

$$B_{2i} = \sum_{k=1}^{NC} (d_k/|S_k|) * (\sum_{j \epsilon S_k} n(i,j)), \text{ integer}$$

$$B_{1i} \quad [(1/2 * B_{2i})]$$

$$B_{3i} = 2*B_{2i}$$

$$f_{0i} = [(p/(1-p))*(s_{1i}*B_{1i})]$$

$$f_{1i} = [p/(1-p)* s_{2i}*(B_{2i} - B_{1i})]$$

$$f_{2i} = [p/(1-p)* s_{3i}*(B_{3i} - B_{2i})]$$

$$p = 0.1, 0.5$$

For our problems "p" represents the percentage of total cost which is a fixed-charge. In other words, a low value of "p" results in the fixed-charges being a relatively small portion of the total cost of each intermediate product. Our calculations are based upon the following principle. When the amount of x_i used is at either B_{1i}, B_{2i}, or B_{3i}, then "p" represents the actual fixed-charge percentage of the total cost. Clearly, at any point other than the three above, and $x_i=0$, the fixed-charge percentage of total cost will be greater than "p" so long as s_{1i}, s_{2i} and s_{3i} are strictly greater than zero.

To ensure each intermediate product is utilized, we eliminate zero columns from matrix n. Notice that the calculation of B_{2i} insures a feasible solution with each x_i at, or near, B_{2i}.

We run problems with as many as 15 intermediate products (NXi=15). While not large problems, they are very difficult to solve.

RESULTS:

Practice has shown the fixed-charge problem to be an extremely troublesome problem to solve using the branch-and-bound technique. Through our sharpness property, we know that the difficulty is caused by the inherently poor "relaxed" approximation, even using the best possible "sharp" representation. Our test problems show that the problems remain difficult even using our sharp representation, and also that there remains an advantage to these representations. Even the "best possible" linear relaxation is evidently fairly inaccurate as an approximation to the fixed-charge functions.

Tables 6-9 represent samples run using Land-Powell's branch-and-bound Mixed Integer code. Problems of Tables 10 and 11 were run using Martin's BANDBX code, and CDC's APEX IV was used for the problems in Table 12.

Our first experiment is reported in Table 6, and does not involve the SEMI-SHARP modelling.

Table 6. Experiments Using Land-Powell, NXi=5.

# SAMPLES	P = 0.5	TIME	# NODES	DISCRETE/ LP RATIO
8	SHARP	9.556	30.75	1.36
8	I	34.04	71.5	1.86
8	II	62.42	81.75	1.36

# SAMPLES	P = 0.1	TIME	# NODES	DISCRETE/ LP RATIO
8	SHARP	25.01	83.38	1.22
8	I	65.61	138.25	1.48
8	II	106.29	131.38	1.22

Table 6 shows that small problems are difficult to solve, and that our combined representation is superior in all three categories. We also notice that, even though the problems with small fixed-charges (p=0.1) have more accurate initial LP values (22% versus 36% using our combined model), the problems are actually harder to solve than those with significant (p = 0.5) fixed-charges.

The degree of fixed-charges does not appear to affect the computational advantage of our combined method (w.r.t. nodes and time).

Aside from the indication that the combined representation is superior to either standard method, it is interesting to note that the Standard II representation appears to give sharp initial LP relaxation solutions. The Discrete/L.P. ratios are identical to those of our sharp representation (the L.P. solutions are identical). But after starting at a much closer solution than Standard I, the total nodes required is virtually the same for both I and II. Furthermore, even if the initial (II) solution is sharp, it requires more nodes than our sharp representation.

In our second experiment for these fixed-charge problems, we retain the better of the two "textbook-like" formulations from the first experiment (this was STANDARD I), and run it together with our SHARP and SEMI-SHARP representations. We also gather more data than before. The results are reported in Tables 7, 8, and 9.

Table 7. Seven Problems with p=0.3, NXi=5
Using Land-Powell.

Formulation	Avg. Time to LP	Avg. Time to Find Optimum	Total Time	Number of Nodes	Discrete/LP
SHARP	1.3 sec.	14.3	18.7	63.3	1.26
I	2.0	8.7	41.3	84.4	1.50
SEMI-SHARP	3.8	30.0	60.9	79.3	1.26

Table 8. Problems with p=0.3, NXi=6,
Using Land-Powell.

Formulation	Avg. Time to LP	Avg. Time to Find Optimum	Total Time	Number of Nodes	Discrete/LP
SHARP	2.0 sec.	22.6	36.5	94.8	1.32
I	3.0	40.3	120.2	176.6	1.56
SEMI-SHARP	5.5	25.2	129.2	119.2	1.32

Table 9. A Hard Problem, p=0.3, NXi=6,
Using Pand-Powell.

Formulation	Time to LP	Time to Opt.	Total Time	Nodes	Discrete/LP
SHARP	2.0 sec.	57.4	62.9	137	1.24
I	3.4	67.7	133	900	1.46
SEMI-SHARP	5.4	113.4	400	unknown	1.24

In Table 7, it is interesting to note that STANDARD I actually outperforms SEMI-SHARP in terms of the "Total Time" criterion (as well as the average time-to-find an optimum). However, as the number of binary variables increase from 30 to 36 (NXi = 5 to NXi = 6), the gap in performance noticeably diminishes. We conjecture that these problems are simply too small for the SEMI-SHARP modelling to have an advantage.

The SHARP modelling is superior in the "Total Time" criterion, although for the set of smaller problems (NXi=5) it is somewhat slower in locating a optimum. The SHARP modelling also has somewhat fewer nodes in the search tree, but this relative advantage is not of consequence. Of course, the SHARP modelling has a better Discrete/LP ratio, but it is far too large to permit heuristic solutions at reasonable levels of error (0-4%) for industrial problems.

A harder problem with NXi=6 is separately reported in Table 9. It is solved to optimality only by the SHARP formulation, although all three formulations do find the optimum.

Tables 10, 11, 12, and 13 compare our SHARP and SEMI-SHARP formulations. Table 10 compares three problems of size NXi=5 using Martin's BANDBX code. The results in Table 10 indicate the dominance of our hereditarily SHARP formulation over the SEMI-SHARP formulation. The SHARP formulation is at least 3 times more efficient in node counts and 8 times faster in total CPU time.

Tables 11, 12, and 13 represent the larger (NXi=10, NXi=15) samples. Table 11 contains the results using our "2% Heuristic," Table

Table 10. Three Problems With NXi=5,
Using BANDBX.

| | Time (CPU seconds) | | | Nodes | | Discrete/LP |
| | To Find | | | To Find | | |
Problem	LP	Discrete	Total	Best Dscrete	Total	Ratio
1 SHARP	0.55	3.19	6.82	15	25	1.38
SEMI-SHARP	1.68	31.2	54.4	56	78	1.38
2 SHARP	0.58	7.05	10.87	26	38	1.35
SEMI-SHARP	1.58	49.7	95.87	81	133	1.35
3 SHARP	0.57	1.74	4.67	10	19	1.23
SEMI-SHARP	1.64	59.49	126.32	82	155	1.23

Table 11.   Six Problems With NXi=10
With 2% Heuristic, using BANDBX.

Problem	Time (CPU seconds)		Nodes		Discrete/LP Ratio
	To Find	Total	To Find	Total	
1 SHARP	52.32	118.82	69	128	1.33
SEMI-SHARP	380.6	600*	177	241*	1.33
2 SHARP	24.2	400*	36	357*	1.31
SEMI-SHARP	405.3	600*	192	268*	1.29
3 SHARP	36.6	267.6	48	229	1.26
SEMI-SHARP	246.0	600*	123	246*	1.26
4 SHARP	22.9	202.8	30	186	1.29
SEMI-SHARP	--	600*	--	--	No Discrete Found
5 SHARP	40.28	305.0	51	276	1.31
SEMI-SHARP	346.1	600*	131	197*	1.31
6 SHARP	9.46	505.5	16	429	1.27
SEMI-SHARP	341.3	600*	116	183*	1.26

*The Branch-and-bound Algorithm automatically stops after the cpu time
indicated to avoid excessive computer changes. Thus, Total Time and
Total Node columns summarize the conditions as the algorithm stopped.

Table 12. Six Problems With NXi=10
Exact Algorithm, Using BANDBX.

Problem	Time (CPU seconds)		Nodes		Discrete/LP Ratio
	To Find	Total	To Find	Total	
1 SHARP	57.37	128.99	74	139	1.33
SEMI-SHARP	544.2	600*	237	251*	1.33
2 SHARP	32.7	600*	42	427*	1.31
SEMI-SHARP	407.3	600*	192	244*	1.29
3 SHARP	37.6	315.5	48	274	1.26
SEMI-SHARP	248.7	600*	123	243*	1.26
4 SHARP	28.83	274.5	30	214	1.29
SEMI-SHARP	528.1	600*	241	262*	1.31
5 SHARP	90.48	346.4	82	266	1.30
SEMI-SHARP	369.12	600*	131	243*	1.30
6 SHARP	22.12	418.1	26	310	1.25
SEMI-SHARP	330.8	600*	135	228*	1.25

*(see previous page)

12 contains identical problems run using an exact algorithm. To read the table, notice that the column entitled "Best Discrete Solution found" may not be the exact optimal solution of the problem. The 2% heuristic may, and often times does fathom a limb of the branch-and-bound tree before the actual exact optimum is found. Correspondingly, the "NODES TO FIND" column indicates the number of nodes required to find the best discrete solution found. The most prominent result of the samples is that the SEMI-SHARP formulation is unable to solve any of the samples to optimality, or even to within 2% of optimality (using the 2% heuristic) within 10 CPU minutes of computer time. The next most prominent feature is the difficulty of the problems even for the SHARP formulation. Notice the high (LP/DISCRETE) ratios. (We look at the relationship between the LP relaxation proximity to the Discrete solution and problem difficulty in the next chapter).

We also find that the 2% heuristic has a very minor effect upon problem solution. While _finding_ the best available discrete solution faster and more efficiently than the exact algorithm, the heuristic algorithm does not always terminate as fast or as efficiently. Detailed computer printouts indicate that when the heuristic causes a branch-and-bound limb to fathom before reaching the exact optimum (problems #2, 5, 6), the algorithm may search more nodes before terminating with a soluton guaranteed to be within 2% of the exact optimum.

Table 13 summarizes three problems run to exact optimality using APEX IV. The first problem is identical to problem #6 of Table 11 and Table 12. Notice that the times are actually APEX Units and cannot be

Table 13. Three Problems Using APEX IV.

Problem	Size NXi=	Time (APEX Units)			Nodes		Discrete/LP Ratio
		LP	To Find	Total	To Find	Total	
1 SHARP	10	2.25	92.0	192.1	137	315	1.25
SEMI-SHARP		4.15	617.0	972.7*	699	2113*	1.25
2 SHARP	10	2.31	90.237	186.9	182	373	1.26
SEMI-SHARP		4.14	150.1	2305*	135	2668*	1.26
3 SHARP	15	3.5	199.6	1458*	229	1791*	1.25
SEMI-SHARP		7.2	615.1	2489*	420	2003*	1.25

*(see previous page)

directly compared to CPU seconds, even though we have found APEX units to be virtually identical to CPU seconds in our experiments. Again, even though SEMI-SHARP produces an identical LP relaxation solution as the SHARP formuation, its (SEMI-SHARP's) lack of hereditary sharpness causes the code to search much longer to even find a feasible integer solution. Since the SEMI-SHARP formulation does not solve the problems, exact comparisons are unavailable. However, the SHARP formulation is at least 7 times more efficient and over 10 times faster (problem #2). The third problem is the largest, NXi=15, and the results display the difficulty of these type problems.

The results of our tests indicate a definite computational advantage of our hereditarily SHARP formulation over both non-sharp and non-hereditarily sharp formulations. The advantage appears to increase with problem size, however the difficulty of the problems prevents us from confirming our hypothesis.

APPENDIX A:

TWO FORMULATIONS FOR BOUNDED-MIP REPRESENTABILITY

Suppose that a finite union S of polyhedra $S = P_1 \cup \ldots \cup P_s$ satisfies the condition β) on recession directions, from the Theorem of Section 1. Then by that Theorem, S is bounded-MIP representable. In fact, in chapter II we provided two mixed-integer formulations for the set S: the "polyhedral" form, and the "extreme point" form. The "polyhedral form" is given as follows. If $P_h = \{x \mid A^h x \geqslant b^h\}$ for $h=1,\ldots,s$ is a polyhedral representation of P_h, then S is represented by

$$x = \sum_{h=1}^{s} x^h \qquad (A1.1)$$

$$A^h x^h \geqslant b^h \lambda_h, \qquad h=1,\ldots,s$$

$$\sum_{h=1}^{s} \lambda_h = 1, \quad \lambda_h \geqslant 0 \qquad \text{for } h = 1,\ldots,s$$

$$\lambda_h \text{ integer}$$

(A1.1) is the form given by Balas [2] for the constraints of a disjunctive program. It is also, after algebraic simplifications, the dual to the linear system of the co-proposition of a statement $\bigvee_{h=1}^{t} (A^h x \geqslant b^h)$ in "disjunctive normal form" (for definition, see [13]) where "V" indicates the "or" connective, or disjunction (for a proof of this fact, see [10]).

Let us consider first a "distribution center" formulation. Here, if any flow x_j from the center ($a \leqslant j \leqslant t$) is positive, the center must be built, and so a "fixed charge" of $c > 0$ assessed. Using the variable z for the charge, we have $t=2$ and $S = P_1 \cup P_2$, where $P_1 = \{(z, x_1, \ldots, x_t) | z \geqslant 0$ and all $x_j = 0\}$, $P_2 = \{(z, x_1, \ldots, x_t) | z \geqslant c$ and all $0 \leqslant x_j \leqslant M_j\}$. Indeed, $S = \text{epi}(f)$ where

$$f(x_1, \ldots, x_t) = \begin{cases} 1 & \text{if any } x_j > 0, \text{ where all } x_j \geqslant 0, \\ & \qquad\qquad \text{and } x_j \leqslant M_j; \\ 0 & \text{if all } x_j = 0 \end{cases} \qquad (A1.2)$$

We seek a modelling of the function f, assuming a minimizing objective function in which $+f$ occurs.

To obain on sharp modelling, we utilize the polyhedral form (A1.1), and we obtain:

$$(z, x_1, \ldots, x_t) = (z^1, x_1^1, \ldots, x_t^1) + (z^2, x_1^2, \ldots, x_t^2) \qquad (A1.2)$$

$$z^1 \geqslant 0 \cdot \lambda_1 \qquad z^2 \geqslant \lambda_2 \cdot c$$

$$x_1^1 = 0 \cdot \lambda_1, \qquad 0 \leqslant x_1^2 \leqslant \lambda_2 \cdot M_1$$

$$\vdots \qquad\qquad \vdots$$

$$x_t^1 = 0 \cdot \lambda_1, \qquad 0 \leqslant x_t^2 \leqslant \lambda_2 \cdot M_t$$

$$\lambda_1 + \lambda_2 = 1, \qquad \lambda_1 \geqslant 0, \ \lambda_2 \geqslant 0$$

$$\lambda_1 \text{ and } \lambda_2 \text{ integer}$$

Upon simplification (which is optional for use in algorithms, but will make

the modelling more perspicuous), we have $x_j^2 = x_j$ for all j, $\lambda_2 \cdot c > z^2 > z$, and (A1.2) gives constraints

$$0 \leqslant x_j \leqslant M_j \cdot z \qquad (A1.3)$$

where $+c \cdot z$ is to be entered in the (minimizing) objective function. We note that (A1.3) is different from the formulation $x_1 + \ldots + x_t \leqslant M \cdot z$, with $M = M_1 + \ldots + M_t$, that is often suggested, and which is not sharp.

The "<u>extreme point</u>" formulation requires that we write each polyhedron P_h as the sum of the convex span of points $v^{1,h}, \ldots, v^{\alpha(h),h}$, which may be dependent on h (and which may be extreme points, for example) and a convex cone generated by elements $w^1, \ldots, w^\beta$ which are to be independent of h. While the condition, that the P_h all have the same recession directions, is not satisfied by all unions $S = P_1 \cup \ldots \cup P_j$ representing S, it is satisfied by some union, when S is bounded-MIP-representable, as noted in Section 3.2.

The extreme point formulation of S is

$$x = \sum_{h=1}^{s} \sum_{i=1}^{\alpha(h)} \lambda_{hi} v^{hi} + \sum_{j=1}^{\beta} \theta_j w^j \qquad (A1.3)$$

$$\lambda_{hi} \geqslant 0, \qquad \theta_j \geqslant 0$$

$$\lambda_h = \sum_{i=1}^{\alpha(h)} \lambda_{hi}$$

$$\sum_{h=1}^{s} \lambda_h = 1$$

all λ_h integer

The extreme point formulation has, generally, fewer constraints but more variables than the polyhedral formulation. From a practical perspective, the extreme point formulation cannot be used when any one of the polyhedra is a "large dimensional cube," or any other polyhedron with a huge number of extreme points. For example, we would not use it in the distribution center problem, with t very much more than t=3 or 4, since P_2 is a hypercube.

In function formulations where $S = epi(f)$ and f is defined only on a _bounded_ domain, the sole recession direction is the (upward) vertical one. Since the function is to be entered into a minimizing objective function, at a minimum the recession direction is not used, so in our formulation work we often ignore it.

We conclude by indicating the derivation of the extreme point formulation of the function of Figure 1. We have $S = epi(f)$ $= P_1 \cup P_2 \cup P_3 \cup P_4$, where the polyhedra and their extreme points are as follows:

Polyhedron	Extreme Points	
$P_1 = \{(z,x) \mid z > 0, x = 0\}$	$(0,0)$	λ_0
$P_2 = \{(z,x) \mid 0 < x < B_{1i},$ $z > f_{0i} + (s_{1i})x\}$	$(f_{0i},0),$ $(f_{0i}+(s_{1i})(B_{1i}), B_{1i})$	λ_{11} λ_{12}
$P_3 = \{(z,x) \mid B_{1i} < x < B_{2i},$ $z > f_{0i} + f_{1i} + (s_{1i})(B_{1i})$ $+(x-B_{1i})(s_{2i})\}$	$(f_{0i} + f_{1i} + (s_{1i})(B_{1i}),(B_{1i}),$ $(f_{0i} + f_{1i} + (s_{1i})(B_{1i})+$ $(s_{2i})(B_{2i}-B_{1i}), B_{2i})$	λ_{21} λ_{22}
$P_4 = \{(z,x) \mid B_{2i} < x < B_{3i},$ $z > f_{0i} + f_{1i} + f_{2i}$ $+(s_{1i})(B_{1i})$ $+(s_{2i})(B_{2i}-B_{1i})$ $+(s_{3i})(x-B_{2i})\}$	$(f_{0i}+f_{1i}+f_{2i}+(s_{1i})(B_{1i})$ $+(s_{2i})(B_{2i}-B_{1i}), B_{2i}),$ $(f_{0i}+f_{1i}+f_{2i}+(s_{1i})(B_{1i})$ $+(s_{2i})(B_{2i}-B_{1i})$ $+(s_{3i})(B_{3i}-B_{2i}), B_{3i})$	λ_{31} λ_{32}

An algebraic computation of the extreme point from (Al.3) from the above data, will yield the formulation (4.6) used in our experiment.

APPENDIX B

ILLUSTRATION OF THE CALCULATION OF UPPER BOUNDS IN THE STANDARD FORMULATION

<u>Sample</u> = 3 divisions
2 technologies/division
3 const./division/tech.
3 common constraints
3 products/division

<u>max:</u>

$$x1 + 9\times2 + 8\times3 + 9\times4 + 10\times5 + 10\times6 + 6\times7 + 4\times8 + 7\times9$$

s.t.:

(common) (1) $8\times1 + 9\times2 + 4\times3 + 2\times4 + 3\times5 + 8\times6 + 1\times7 + 8\times8 + 8\times9 \leqslant 103*$

(2) $5\times1 \quad + 3\times3 + 7\times4 + 6\times5 \quad + 6\times7 + 8\times8 + \times9 \leqslant 73*$

(3) $1\times1 + 3\times2 + 3\times3 + 3\times4 + 5\times5 + 7\times6 + 7\times7 + 9\times8 + 7\times9 \leqslant 91*$

Division 1

$$\begin{array}{l} (4)\ \ 8\times1 + 5\times2 + 7\times3 \leqslant 41 \\ (5)\ \ 4\times1 + 3\times2 + 7\times3 \leqslant 29 \\ (6)\ \ 2\times1 + 4\times2 + 9\times3 \leqslant 31 \end{array}$$
or
$$\begin{array}{l} (7)\ \ 7\times1 + 1\times2 + 8\times3 \leqslant 33** \\ (8)\ \ 8\times1 + 2\times2 + 8\times3 \leqslant 37 \\ (9)\quad\quad 7\times2 + \times3 \leqslant 17 \end{array}$$

$$\begin{array}{l} (10)\quad\quad\quad 8\times5 + 7\times6 \leqslant 27 \\ (11)\ \ 4\times5 + 2\times5 + 6\times6 \leqslant 19 \\ (12)\ \ 1\times4 + 6\times5 + 2\times6 \leqslant 35 \end{array}$$
or
$$\begin{array}{l} (14)\ \ 5\times4 + 8\times5 + 4\times6 \leqslant 35 \\ (15)\ \ 3\times4 + 2\times5 + 6\times6 \leqslant 23 \\ (16)\ \ 8\times4 \quad 3\times5 + 3\times6 \leqslant 29 \end{array}$$

$$\begin{array}{l} (18)\ \ 4\times7 \quad 8\times8 + 7\times9 \leqslant 39 \\ (19)\ \ 6\times7 + 8\times8 + 3\times9 \leqslant 35 \\ (20)\ \ 5\times7 + 8\times8 + 6\times9 \leqslant 29 \end{array}$$
or
$$\begin{array}{l} (21)\ \ 3\times7 + 8\times8 \quad\quad \leqslant 23 \\ (22)\ \ 8\times7 + 3\times8 + 8\times9 \leqslant 39 \\ (23)\ \ 5\times7 \quad 3\times8 + 3\times9 \leqslant 23 \end{array}$$

*   r.h.s. = $\alpha*$(sum coeff) + 1, here α = 2.

** r.h.s. = 2 (sum of coeff) + 1.

For our example, we can put these bounds in Division 1:

j=1

$$x(1) = \min \left\{ \left\lfloor \frac{41}{8} \right\rfloor, \left\lfloor \frac{29}{4} \right\rfloor, \left\lfloor \frac{31}{2} \right\rfloor \right\} = 5 \quad R(1) = \max \{5, 4\} + 1 = 6$$

$$x(2) = \min \left\{ \left\lfloor \frac{33}{7} \right\rfloor, \left\lfloor 37/8 \right\rfloor, +\infty \right\} = 4$$

j=2

$$x(1) = \min \left\{ \left\lfloor \frac{41}{5} \right\rfloor, \left\lfloor \frac{29}{3} \right\rfloor, \left\lfloor \frac{31}{4} \right\rfloor \right\} = 7 \quad R(2) = \max \{7, 2\} + 1 = 8$$

$$x(2) = \min \left\{ \left\lfloor \frac{33}{1} \right\rfloor, \left\lfloor 37/2 \right\rfloor, \left\lfloor 17/2 \right\rfloor \right\} = 2$$

j=3

$$x(1) = \min \left\{ \left\lfloor \frac{41}{7} \right\rfloor, \left\lfloor \frac{29}{7} \right\rfloor, \left\lfloor \frac{31}{9} \right\rfloor \right\} = 3 \quad R(3) = \max \{3, 4\} + 1 = 5$$

$$x(2) = \min \left\{ \left\lfloor 33/8 \right\rfloor, \left\lfloor 37/8 \right\rfloor, \left\lfloor 17/1 \right\rfloor \right\} = 4$$

The Upper Bounds are:

UB(4): 8(6) + 5(8) + 7(5) = 123   UB(6): 2(6) + 4(8) + 9(5) = 89

UB(5): 4(6) + 3(8) + 7(5) = 83   UB(7): 7(6) + 1(8) + 8(5) = 90, etc.

In the Standard Form, the Division Constraints become:

Division 1

(1) $8x_1 + 5x_2 + 7x_3 + 82z_1^{**} \leqslant 123$ (4) $7x_1 + 1x_2 + 8x_3 + 57z_2^{**} \leqslant 90$

(2) $4x_1 + 3x_2 + 7x_3 + 54z_1 \leqslant 83$ (5) $8x_1 + 2x_2 + 8x_3 + 67z_2 \leqslant 104$

(3) $2x_1 + 4x_2 + 9x_3 + 58z_1 \leqslant 89$ (6) $\quad + 7x_2 + 1x_3 + 44z_2 \leqslant 61$

(7) $z_1 + z_2 = 1$

where ** coefficients of discrete variables are found as below:

$$82 = (\text{Upper Bound} - \text{R.H.S.}) = (123 - 41) = 82$$

$$(83 - 29) = 54$$

$$(89 - 31) = 58, \text{ etc.}$$

REFERENCES

1. E. Balas, "Disjunctive Programming: Cutting-planes from Logical Conditions," in O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Nonlinear Programming 2, Academic Press, New York (1975), pp. 279-312.

2. E. Balas, "Disjunctive Programming: Facets of the Convex Hull of Feasible Points," no. 348, GSIA, Carnegie-Mellon University, 1974.

3. C. E. Blair and R. G. Jeroslow, "A Converse for Disjunctive Constraints" Journal of Optimization Theory and Its Applications 25 (1978), pp. 195-206.

4. H. Crowder, E. L. Johnson, and M. W. Padberg, "Solving Large-Scale Zero-One Linear Programming Problems," IBM Research Report RC8888, 1981.

5. S. Bradley, A. Hax and T. Magnanti, Applied Mathematical Programming, Addison-Wesley Pub. Co., Reading, Mass., 1977.

6. G. B. Dantzig, Linear Programming and Extensions, Princeton, New Jersey, Princeton University Press, 1963.

7. G. D. Eppen and F. J. Gould, Quantitative Concepts for Management, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1979.

8. A. M. Geoffrion and G. W. Graves, "Multicommodity Distribution System Design by Benders Decomposition," Management Science 20 (1974), pp. 822-844.

9. F. Glover, "New Results on Equivalent Integer Programming Formulations," Mathematical Programming 8 (1975), pp. 84-90.

10. F. Glover, "Polyhedral Annexation in Mixed Integer and Combinatorial Programming," Mathematical Programming 9 (1975), pp. 161-188.

11. A. C. Ho, "Cutting-planes for Disjunctive Programs: Balas' Aggregated Problem," Carnegie-Mellon University, October 1976 (a Ph.D. student summer research paper).

12. T. Ibaraki, "Integer Programming Formulation of Combinatorial Optimization Problems," Discrete Mathematics 16 (1976), pp. 39-52.

13. R. Jeroslow, "Cutting-planes for Relaxations of Integer Programs," no. 347, GSIA, Carnegie-Mellon University, 1974.

14. R. Jeroslow, "Cutting-plane Theory: Disjunctive Methods," Annals of Discrete Mathematics 1 (1977), pp. 293-330.

15. R. Jeroslow, "Representations of Unbounded Optimizations as Integer Programs," Journal on Optimization Theory and Its Applications 30 (1980), pp. 39-351.

16. R. Jeroslow and J. Lowe, "Modelling with Integer Variables," 1981.

17. C. B. Krabek, "Some Experiments in Mixed Integer Matrix Reduction," Control Data Corporation, presented at ORSA/TIMS Hawaii Meeting, 1979.

18. A. Land and S. Powell, Fortran Codes for Mathematical Programming: Linear, Quadratic and Discrete, John Wiley & Sons, London, England, 1973.

19. R. R. Meyer, "Integer and Mixed Integer Programming Models: General Properties," Journal on Optimization Theory and its Applications 16 (1975), pp. 191-206.

20. R. R. Meyer, "Mixed-Integer Minimization Models for Piecewise-Linear Functions of a Single Variable," Discrete Mathematics 16 (1976), pp. 163-171.

21. R. R. Meyer, "A Theoretical and Computational Comparison of 'Equivalent' Mixed Integer Formulations," Naval Research Logistics Quarterly 28 (1981), pp. 115-131.

22. R. R. Meyer and M. V. Thakkar, "Rational Mixed Integer Minimization Models," MRC #1552, University of Wisconsin, 1976.

23. L. A. Oley and R. J. Sjoquist, "Automatic Reformulation of Mixed and Pure Integer Models to Reduce Solution Time in Apex IV," Control Data Corporation, presented at ORSA/TIMS San Diego Meeting october 1982.

24. D. C. Sommer, "Computational Experience with the Ophelie Mixed Integer Code," Control Data Corporation, presented at ORSA/TIMS, 1970.

25. H. P. Williams, Model Building in Mathematical Programming, John Wiley and Sons (Wiley Interscience), 1978.

26. R. D. Young, "Hyperclindrically-deduced Cuts in Zero-One integer Programs," Operations Research 19 (1971), pp. 1393-1405.

CHAPTER IV

ONE TEST OF THE PROXIMITY OF THE LINEAR RELAXATION
AS A GUIDE TO PROBLEM DIFFICULTY

Introduction

The purpose of this chapter is to add further confirmation to a

commonly accepted viewpoint, that the proximity of the linear relaxation

is a gauge of problem difficulty.

From our earlier results we see that the Multi-Divisional

problems, whose LP relaxation optimum is within 1 percent of the integer

optimum, are much easier to solve than the fixed-charge problems whose

LP relaxation optimum is not within 25 percent of the integer optimum.

In this chapter we present two classes of problems, of similar

construction, whose LP relaxations are very dissimilar in terms of their

proximity to the convex span of the integer solutions. We then compare

solution difficulty of the two problems using Martin's BANDBX system.

The first class of problems is a slight variation of the well-

known fixed-charge problem. The second class of problems is of the

fixed-benefit variety, in which a benefit is received if a product is

utilized.

Both problem sets involve "minimum usage levels" which activate

the fixed-charge or benefit, as well as upper bounds on the value of the

variable. These two parameters determine how close the LP relaxation is

to the MIP formulation. In the 'usual' setting of these parameters, the

usage level is much smaller than the upperbound. For usual scenarios,

the fixed-charge LP relaxation feasible set is much larger than the feasible set of the MIP formulation, even when the best possible (i.e. sharp) formulation is used. On the other hand, the LP relaxation feasible set for the fixed-benefit problem is usually nearly identical to its integer feasible set. However, by altering the minimum usage level so that it is much larger than usual, the fixed-benefit LP relaxation feasible set becomes noticeably larger than its MIP feasible set. Thus we are able to test how problem difficulty varies as the differences between the LP relaxation feasible sets and MIP feasible sets are altered. Since we proved earlier that the LP relaxation feasible set of any SHARP formulation is exactly the closure of the convex hull of the MIP representable set S, we are, therefore, testing the effect on problem difficulty as the size of the convex hull of a MIP representable set changes.

We remark that we use both minimum usage levels and upper bounds in our problems, since the fixed charge problems require bounds, while the fixed-benefit problems require usage levels, in order to be formulated as MIPs. By giving both problem sets both usage levels and bounds, they are more comparable.

Problems

The general problem description for both the fixed-charge problem (FC) and the fixed-benefit problem (FB) have the mathematical form,

$$\text{min:} \quad \sum_{j=1}^{n} \left(c_j x_j + f_j(x_j) \right) \qquad (4.1)$$

$$\text{s.t.:} \quad \underline{A} \underline{x} \geq \underline{b}$$

$$\underline{x} \geq 0$$

where $f_j(x_j)$ is a nonlinear function defined as follows. For the fixed-charge case,

$$\text{(FC)} \quad f_j(x_j) = \begin{cases} 0 & \text{if } x_j = 0 \\ s_j & \text{if } \delta_j \leq x_j \leq M_j \end{cases} \quad \text{for all } j \qquad (4.2)$$

and for the fixed-benefit problem,

$$\text{(FB)} \quad f_j(x_j) = \begin{cases} 0 & \text{if } x_j = 0 \\ -s_j & \text{if } \delta_j \leq x_j \leq M_j \end{cases} \quad \text{for all } j. \qquad (4.3)$$

In both (4.2) and (4.3), s_j is nonnegative, δ_j is strictly greater than zero, and M_j is greater than δ_j.

A typical example is the product mix problem in which the objective is to minimize the cost of resources while satisfying certain quality-control, or demand constraints. The cost of the resources involves a linear portion $(c_j x_j)$ as well as the nonlinear $f_j(x_j)$ term. In the FC case, an additional charge of s_j is incurred if the amount of resource $j(x_j)$ falls within the $[\delta_j, M_j]$ interval. In this case, δ_j is the minimum usage level and M_j is the maximum usage level. In the FB case, a benefit, or negative cost $(-s_j)$ is received if resource j falls within

the $[\delta_j, M_j]$ interval. Note that a value of x_j strictly between 0 and δ_j is not permitted.

Defining the set S_j as $\text{epi}(f_j(x_j))$, and $S = \bigcup_{j=1}^{n} S_j$, we see that S is MIP representable. The graph of S_j for both problems follows.
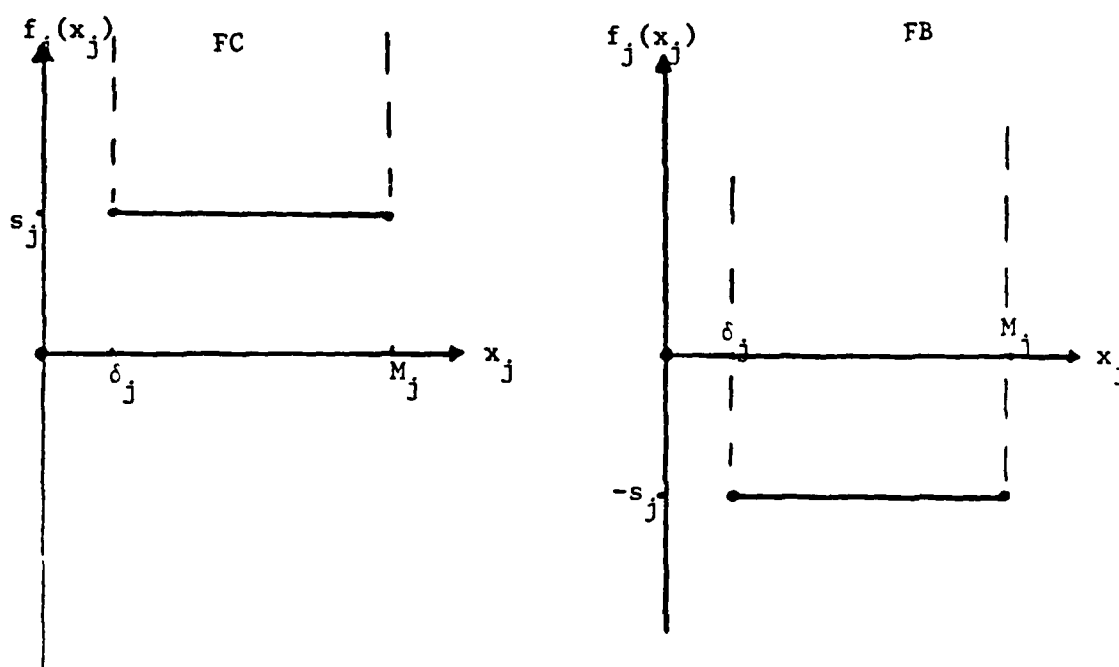
Figure 4.1
Fixed-Charge vs. Fixed-Benefit

Using the "extreme point" sharp modelling for the FC problem we add to the constraints in (4.1),

$$x_j = \Theta_{1j}\, \delta_j + \Theta_{2j} M_j \tag{4.5}$$

$$\text{(FC)} \quad \lambda_j = \Theta_{1j} + \Theta_{2j}$$

$$\lambda_j \in \{0, 1\}, \; \Theta_{1j}, \; \Theta_{2j} \geq 0$$

and replace $f_j(x_j)$ in the objective function with the term $\lambda_j^* s_j$. The modelling in the FB case is identical except $(-\lambda_j^* s_j)$ is used in the objective function. Thus we have two very similar modellings whose LP relaxations are complementary in that the closer the proximity of the FB LP relaxation to the original MIP set, the worse the proximity of the FC LP relaxation is to its orginal MIP set. A picture best describes this notion.
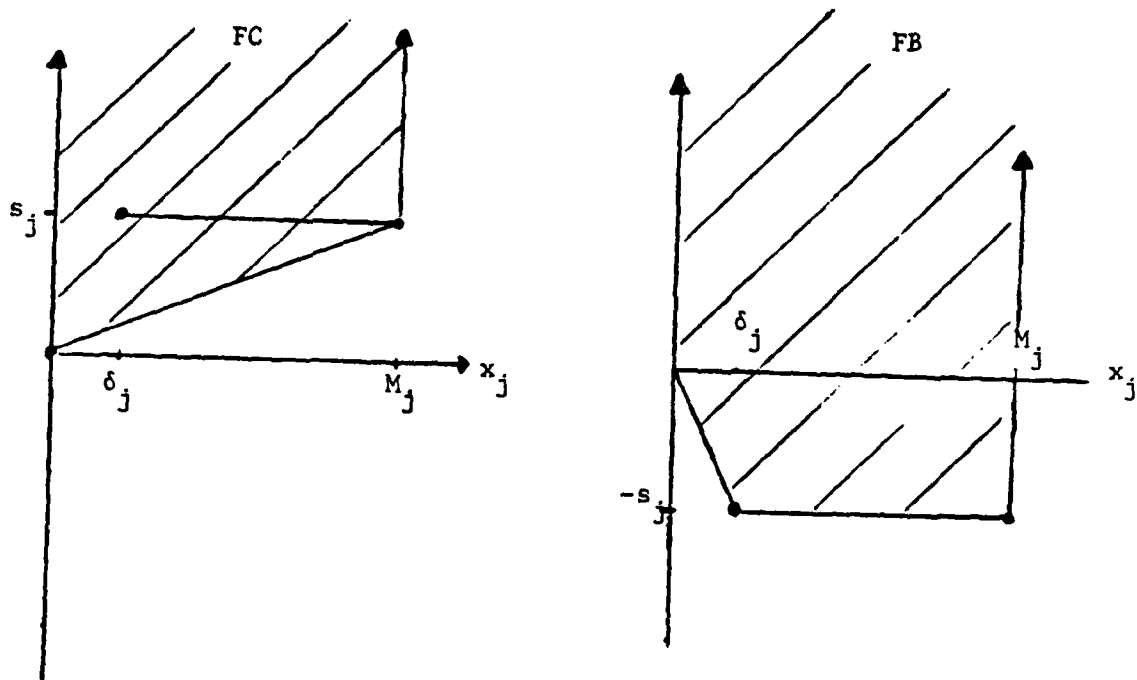
Figure 4.2

L.P. Relaxations (Convex Hulls)

Notice the shaded area representing the LP relaxations of both problems.
First notice that the difference between the original representable set
S_j (4.4) and the LP relaxation (4.6) is much greater for the FC problem
than the FB problem. Thus, if our conjecture is correct, the typical
fixed charge problem is more difficult than the typical fixed benefit
problem. Suppose we depart from the typical case in which S_j is small
with respect to M_j, and increase the value of the minimum usage level to
one-half of the maximum usage level. The LP relaxations now become,
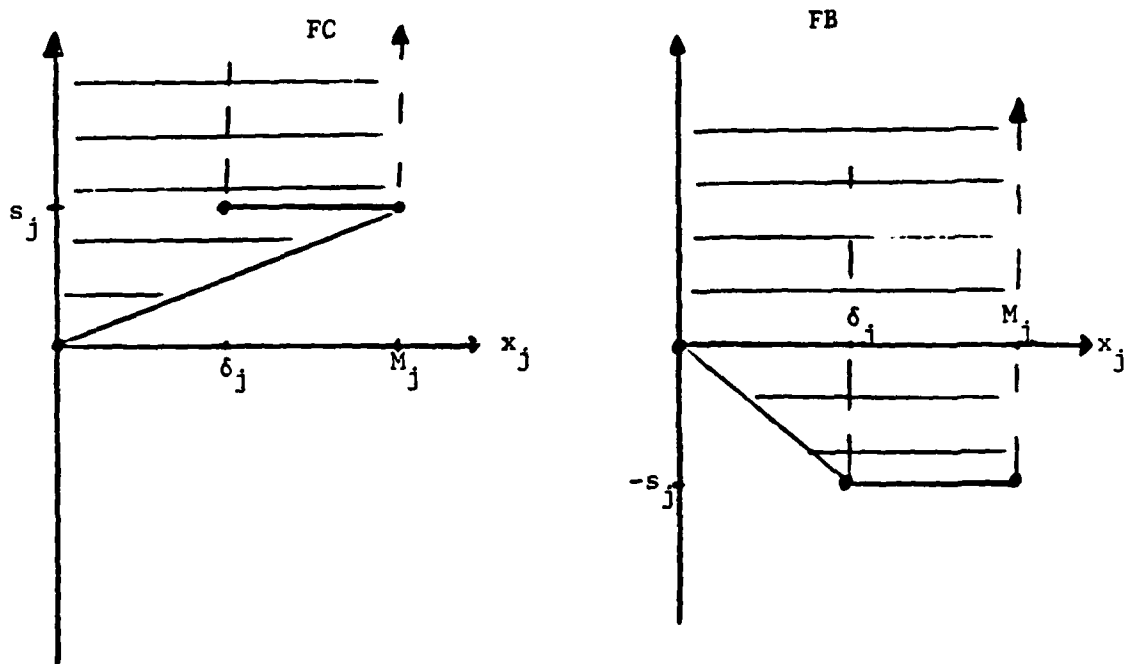
Figure 4.3
Effect of Minimum Usage Level

Notice, the FC LP relaxation is unchanged while the FB LP relaxation does change. More importantly the set $L_j = (clconv(S_j) - S_j)$, while larger in both cases as δ_j increases, its size increase is larger in the FB problem. Therefore we expect that as δ_j increases, the computational advantage of the FB problem may decrease. Our tests confirm our expectations.

Experiments

We run several samples of the problems described by (4.1), (4.2), and (4.3) and modelled using the "extreme point" formulation (4.5). We solve two different sizes of problems, the first (I) contains 10 original variables and 20 original constraints (i.e. A matrix is 20 × 10), the second (II) contains 20 variables and 40 constraints (A matrix is 40 × 20). Using the extreme point formulation, the problems of size I, have 50 constraints and 80 variables, 20 of which are binary; size II problems have 100 constraints, 160 variables, of which 40 are binary variables.

For each problem we randomly choose each a_{ij} as an integer between 1 and 10. Each b_i, $(i=1,n)$ is selected as $b_i = \sum_{j=1}^{n} a_{ij}$, thus $x_j = 1$ for all j, is a feasible solution. Next, we randomly choose each c_j as an integer between 5 and 10. For some problems, the fixed-charge (benefit) s_j is 30% of the total cost of setting a variable to one, i.e. $s_j = 0.3(s_j+c_j)$. For other problems the percentage of the fixed-charge (benefit) is increased to 50% of the total. The maximum usage level (M_j) is 4.0, while the minimum usage level (δ_j) varies between $\delta_j = 0.1$, $\delta_j = 1.0$, and $\delta_j = 2.0$. Thus, the minimum usage level varies from 2.5% to 50% of the maximum usage level.

After the problem parameters are thus randomly chosen, we create a pair of problems. In the first member of each pair, the parameters are used to construct a fixed-charge problem as described above. The second member of each pair is made into a fixed-benefit problem with exactly the same data.

Along with the usual statistics, we compute the ratio of the fixed-Benefit/Fixed-Charge solution times. The results are summarized in Tables 14 through 16. Table 14 summarizes problems of size I and II, with a minimum usage level of 2.0, and a fixed cost prcentage of 30%. Table 15 represents problems of size I and II with minimum usage levels of 1.0 and fixed cost percentage of 30%. Table 16 summarizes problems of both sizes with minimum usge levels of 0.1 and a 50% fixed cost percentage.

As we predict, the FB problem solves faster and more efficiently than the FC problems. But the results are not as dominant as one might expect. First, notice that the (LP/Best Discrete Solution) ratio is over 0.9 for both FB and FC, thus the problems are "easy" fixed-cost problems to begin with. Problems with lower ratios as in Chapter III may produce more dramatic results. From Tables 13 and 14 we find that when the minimum usage level is significant with respect to the upper bound, the FB problems are not much faster, but are more efficient. In fact, in some instances, the FC problem solves faster even though it requires more nodes to solve the problem than its FB counterpart. As our Fig. 4.3 suggests, problems with small minimum usage levels are easily solved as FB problems (see Table 15).

An interesting discovery among all samples is that the FB representation is harder to solve as a linear program than the fixed charge problem. We believe this is partially due to the fact that the c_j values are at least as great as the z_j coefficient (s_j) values. Therefore in the FC relaxation problem, once the $Ax \geq b$ constraints are satisfied then the λ_j variables are set to $\lambda_j = x_j/4$. But for the FB problem, the code will initially try to set all λ_j variables to their upper limit of one, to obtain the benefits, which forces the x_j values to be at least the minimum usage level. But, since increasing the x_j variables adds the relatively large c_j costs, the code then re-adjusts the λ_j values, which tends to increase the number of iterations in the LP problem.

Table 14. Fixed-Charge vs. Fixed-Benefit
$\delta_j = 2.0$, % Fixed-Cost = 30%
Using Martin's BANDBX Code

Problem Size (#)	Solution Time (CPU)				NODES SOLVED				LP/Discrete Solution Ratio	
	FC		FB		FB/FC Ratio Time	FC		FB		FC	FB
	LP	Total	LP	Total		To Find	Total	To Find	Total		
I (5)	1.44	31.6	2.71	24.2	0.76	31	50	17	30	.952	.979
II (2)	8.55	600*	27.5	590.3**	?	152	206[1]	138	181[1]	.970[2]	.988[2]

*neither problem was solved within 600 CPU seconds

**only one of the two problems solved within 600 CPU seconds

[1]Total Nodes unknown as problems were not completely solved.

[2]Best Discrete Solution found at time of shutoff is used in ratio.

Table 15. Fixed-Charge vs. Fixed-Benefit
δ_j = 1.0, % Fixed-Cost = 30%
Using Martin's BANDBX

Problem Size (#)	Solution Time (CPU)				NODES SOLVED				LP/Discrete Solution Ratio	
	FC		FB		FB/FC Ratio Time	FC		FB		FC	FB
	LP	Total	LP	Total		To Find	Total	To Find	Total		
I (3)	1.43	23.1	3.37	9.42	0.41	32	37	8	9	0.961	0.995
II (2)	8.35	337.1	23.8	208.3	0.62	67	117	43	52	0.976	0.997

Table 16. Fixed-Charge vs. Fixed-Benefit
δ_j = 0.1, % Fixed-Cost = 50%
Using Martin's BANDBX

Problem Size (#)	Solution Time (CPU)				NODES SOLVED				LP/Discrete Solution Ratio	
	FC		FB		FB/FC Ratio Time	FC		FB		FC	FB
	LP	Total	LP	Total		To Find	Total	To Find	Total		
I (3)	1.23	51.2	2.81	2.93	0.06	89	95	1	1	0.933	1.0
II (2)	6.92	432.4	24.6	24.8	0.06	81	167	1	1	0.969	1.0
II (2)	6.92	432.4	24.6	24.8	0.06	81	167	1	1	0.969	1.0

CHAPTER V

TWO TESTS OF PROPOSITIONAL LOGIC PROBLEMS

Introduction

The purpose of this chapter is to explore the effectiveness of our Sharp modellings, and "standard" modellings, in problems of propositional logic (see, e.g. [2] or [35] for a discussion of this logic).

The first test is entitled "Satisfiability Testing." In this experiment, we are given a formula in conjunctive normal form. (i.e. as a conjunction of a disjunction of literals - a literal being a propositional letter or its negation) We wish to determine whether the formula is satisfiable.

The given information for the second experiment is a set of implications, $K_i \rightarrow L_i$, for $i=1,\ldots,n$ where each L_i is a disjunction of literals, and K_i is a conjunction of literals in some experiments, and a disjunction of literals in others. We wish to determine whether the set of implications imply that a particular literal is true, provided certain other literals are all given as true. The literals last mentioned are part of the data of the problem.

Our interest in satisfiability testing lies in its role as an important problem type for theoretical analysis, particularly the NP-completeness theory. Our interest in implication testing lies in the use

of implications in the "expert systems" of artificial intelligence [14, 15].

In forcing certain letters $A_1, \ldots, A_t$ to be true and a letter B to be false, with all implications $K_i \rightarrow L_i$ $(i=1,\ldots,n)$ true, and determining if the overall result is consistent, we are in fact ascertaining if B is forced to be true when the "data" on $A_1, \ldots, A_t$ are as forced and the "production rules" $K_i \rightarrow L_i$ $(i=1,\ldots,n)$ are valid. In actual expert systems, more complex relationships are modelled than those which can be represented by propositional logic, but it is the expert system paradigm which motivates us.

In the problem studied here, the MIP code is used only as a consistency tester (i.e., there is no criterion function).

Both propositional logic experiments are MIP representable, in fact, both can be modelled as pure integer programs. However, it is unnecessary to actually declare all variables as integer, thus the problems we solve are mixed-integer programs. One of our initial goals was to develop an efficient branch-and-bound algorithm to solve these problems, but we are surprised to find that standard branch-and-bound code (Martin's BANDBX and APEX IV) solve the problems very effectively. In fact, as shown later, we are forced to alter our random generator to artificially create difficult problems in order to compare the computational effort of different modellings used.

Both modellings solve the problems so efficiently that we stop comparing modellings and concentrate solely on creating difficult problem instances. I.e., for satisfiability problems of the size we

study, there is no advantage to the newer methods, since random
propositional formula are easy to decide by branch-and-bound.

Problems

The propositional logic formulas for the satisfiability
experiments are developed in the following manner. Given the total
number of literals to choose from, ℓ, the number of literals per clause,
s (s < ℓ), and the total number of clauses in the formula, c, we randomly
generate c clauses of size s in the ℓ literals and test whether the
resulting formula is satisfiable. For instance, if $\ell=5$, s=2, and c=3,
the following formula may be randomly generated,

$$K = (A_2 \lor \sim A_5) \land (A_1 \lor A_5) \land (A_1 \lor A_2) \land (A_4 \lor A_5), \qquad (5.1)$$

where "$\lor$" indicates "or," "$\land$" indicates "and", also $\sim A_i$ implies negation
of the literal A_i. To model the formula as a mixed-integer program, we
assign binary variables to each literal of the formula such that if A_i is
true then its associated variable $x_i = 1$, similarily if A_i is false then
$x_i = 0$. Negation of literals, $\sim A_i$, are represented as $1-x_i$. To model
(5.1) we first model the individual clauses, i.e. $(A_1 \lor A_2 \ldots \lor A_3)$.

Modelling the V-connective

For an expression of the form

$$A = A_1 \lor A_2 \lor \ldots A_t, \qquad (5.2)$$

where each A_j has been assigned a variable x_j and a variable x is
assigned to A, a non-sharp modelling follows:

$$x \geqslant (x_1 + x_2 + \ldots + x_t)/t \qquad (5.3)$$
$$x \leqslant x_1 + x_2 + \ldots + x_t.$$

Our Sharp modelling of the expression is;

$$x \geqslant x_j \qquad \text{for } j=1,t \qquad (5.4)$$
$$x \leqslant x_1 + x_2 + \ldots + x_t.$$

The non-sharp model (5.3) is similar to some representations which have been used (largely unsucessfully) in practice.

The model (5.4) which we are calling "Sharp" here is similar to one used successfully in practice in a setting where the logical constraints were part of a much larger group of constraints [Johnson, et al., [31], and has the same character as the "disaggregated" constraints for a distribution center with a fixed-charge [Graves & Geoff., 19]. In both settings it is crucial to successful implementation.

We call (5.4) "sharp" by extension of terminology. The set S to be modelled has the form $S = P_1 \cup S'$, where $P_1 = \{x, x_1, \ldots, x_t)|$ $x = x_1 = \ldots = x_t = 0\}$ is a polyhedron and $S' = \{(x, x_1, \ldots, x_t \mid x = 1,$ $\Sigma x_j \geqslant 1, \text{ all } x_j \in \{0,1\}\}$ is a bounded-MIP representable set (in fact, the union of 2^{t-1} polyhedra). We relax S' to the polyhedron $P_2 = \{(x, x_1, \ldots, x_t) \mid x = 1, \Sigma x_j \geqslant 1, 0 \leqslant x_j \leqslant 1 \text{ all } j\}$ (note that that $P_2 = \text{conv}(S')$) and then we model $P_1 \cup P_2$ by the polyhedral method as follows:

$$x = x^{(1)} + x^{(2)} \tag{5.5}$$

$$x_j = x_j^{(1)} + x_j^{(2)} \quad \text{all } j$$

$$x^{(1)} = 0 \cdot \lambda_1 \qquad\qquad x^{(2)} = 1 \cdot \lambda_2$$

$$x_j^{(1)} = 0 \cdot \lambda_1 \quad \text{all } j \quad \sum_j x_j^{(2)} \geqslant 1 \cdot \lambda_2$$

$$0 \leqslant x_j^{(2)} \leqslant 1 \cdot \lambda_2 \quad \text{all } j$$

upon simplification, we have $x = 0 + \lambda_2 = \lambda_2$, $x_j = 0 + x_j^{(2)} = x_j^{(2)}$, and we note that $x_j = x_j^{(2)} \leqslant \lambda_2 = x$. This gives (5.4) which is sharp for the given <u>relaxation</u> $P_1 \cup P_2$ of $S = P_1 \cup S'$.

All literal variables (x_j) are declared binary, while the formula variable (x) is not declared integer even though it is forced to integer values. Notice our Sharp modelling involves $(t+1)$ constraints while the non-sharp model requires only 2 constraints per expression.

<u>Modelling the "Λ" connective</u>. For an expression of the form:

$$A = A_1 \land A_2 \land \ldots \land A_t \tag{5.6}$$

where each literal has already been assigned a binary variable x_j, and A is assigned a variable x, the model is developed in the following manner. Since (5.6) is equivalent to

$$\sim A = \sim A_1 \ V \sim A_2 \ V \ \ldots \ V \sim A_t ,$$

we use the "or" connective modellings replacing x with $(1-x)$ and x_j with $(1-x_j)$. The nonsharp model is:

$$\sum_{j=1}^{t} x_j > tx \tag{5.7}$$

$$-x + \sum_{j=1}^{t} x_j < (t-1). \tag{5.8}$$

The corresponding Sharp model is,

$$x < x_j \quad \text{for } j=1,t \tag{5.9}$$

$$-x + \sum_{j=1}^{t} x_j < (t-1).$$

The sharp modellings (5.4) and (5.9) are based upon our polyhedral fcrm mentioned in Chapter II. To clarify this modelling, the expression (5.2) can be expressed as,

$$x = 0 \qquad x = 1$$

$$x_1 = 0 \quad \text{or} \quad 0 < x_j < 1 \quad \text{for all } j \tag{5.10}$$

$$\vdots \qquad \qquad \sum_{j=1}^{t} x_j > 1$$

$$x_t = 0$$

with all variables non-negative. Assigning λ_1 as the multiplier for the left side of (5.10) and λ_2 as the right side multiplier, our polyhedral form for MIP representations simplifies to:

$$x = \lambda$$

$$x_j \leqslant \lambda \qquad j=1,\ldots t$$

$$\sum_{j=1}^{t} x_j \geqslant \lambda$$

$$\lambda \in \{0,1\},$$

which simplifies to (5.4).

Combining the modellings of the "V" connective and the "Λ" connective allows us to model conjunctive normal form formulas and, in fact, any propositional expression in 'V', "Λ," and '~.' We next illustrate this with the formula (5.1)

The non-sharp representation is,

$$x_2 + (1-x_5) \geqslant x_6 \qquad (5.11)$$

$$x_2 + (1-x_5) \leqslant 2x_6$$

$$x_1 + x_2 \geqslant x_7$$

$$x_1 + x_2 \leqslant 2x_7$$

$$x_4 + x_5 \geqslant x_8$$

$$x_4 + x_5 \leqslant 2x_8$$

$$x_6 + x_7 + x_8 \geqslant 3x_9$$

$$-x_9 + x_6 + x_7 + x_8 \leqslant 2$$

x_1, x_2, x_4, x_5 are binary

all variables $\geqslant 0$.

In the above, x_6, x_7 x_8 are variables assigned to each of the clauses of the formula, and x_9 is assigned to K. Therefore, once the constraint $x_9 = 1$ is added to the constraint set (5.11), the formula (5.1) is satisfiable if and only if the constraint set is consistent.

Modelling the "$\rightarrow$" Implication. The implication testing experiment requires all implications $(K_i \rightarrow L_i)$ to be true, with $(K_i \rightarrow L_i)$ equivalent to $(\sim K_i \vee L_i)$. Therefore each implication is modelled as an "$\vee$" connective. We next work an example.

The implication

$$(\sim A \wedge B \wedge \sim C) \rightarrow \sim D \qquad (5.12)$$

is transformed to the disjunction

$$A \vee \sim B \vee C \vee \sim D \qquad (5.13)$$

(since $(\sim A \wedge B \wedge \sim C)$ is $(A \vee \sim B \vee C)$). Then (5.13) is modelled as,

$$x_1 + (1-x_2) + x_3 + (1-x_4) \geq 1 \qquad (5.14)$$

and $[x_1 + (1-x_2) + x_3 + (1-x_4)]/4 \leq 1$

Since the implication is required to be true, in (5.14) we have

substituted '1' for the variable x which would (in a setting of nested

logical connectives) have been assigned to the entire subformula in

(5.13).

The second constraint in (5.14) is equivalent to $x_1 - x_2 + x_3 - x_4$

≤ 2, which is redundant for $0 \leq x_j \leq 1$, so it can be dropped. If (5.9)

is used instead of (5.3) to model (5.13), the second constraint in

(5.14) becomes four constraints, specifically: $1 \geq x_1$, $1 \geq 1 - x_2$, $1 \geq$

x_3, $1 \geq 1 - x_4$. Each of these is redundant, and can be dropped.

Therefore, in this example, due to simplifications both (5.3) and (5.4)

yield the same modelling, i.e. the first constraint in (5.14), called a

"generalized set covering constraint," Many of our runs are done via

such constraints, with no difference between (5.3) and (5.4). By using

this modelling, with all propositional letters declared binary, we have

one constraint per implication. In effect, we transform our logic

problem into a generalized set covering problem.

In some other experiments reported below, there will be a

difference between (5.3) and (5.4) formulations.

For example, to model the implication

$$(A \lor B \lor \sim C) \to D \qquad (5.15)$$

we reduce it to

$$(\sim A \wedge \sim B \wedge C) \vee D$$

(as $\sim (A \vee B \vee \sim C)$ is $(\sim A \wedge \sim B \wedge C)$, which has a nested structure. We introduce a binary variable x for the subformula $\sim A \wedge \sim B \wedge C$, and the modelling via (5.3), (5.7) becomes

$$3x \leq (1-x_1) + (1-x_2) + x_3 \qquad (5.16)$$

$$-x + (1-x_1) + (1-x_2) + x_3 \leq 2$$

$$1 \geq (x+x_4)/2$$

$$1 \leq x + x_4$$

The modelling via (5.4), (5.9) becomes

$$x \leq 1 - x_1 \qquad (5.17)$$

$$x \leq 1 - x_2$$

$$x \leq x_3$$

$$1 \geq x$$

$$1 \geq x_4$$

$$1 \leq x + x_4$$

In (5.16), the third constraint is redundant and in (5.17) the third and fourth constraints are redundant. However, even with simplifications, we obtain different representations.

In general, if what preceeds the implication (i.e. K_i in $K_i \rightarrow L_i$) is a conjunction of literals, simplifictions result in the same

generalized set covering problem for both sharp and nonsharp formulations. If disjunctions occur there, even after simplifications, the formulations are different.

In our experiments, we try runs that included the simplifications, some of which are the same (i.e. generalized set covering) for both the sharp and nonsharp representations, and some of which are different. In addition, in some runs we do not make the simplifications which are possible and deliberately left redundant constraints in the program, simply to see what happens when these methods are used "blind."

Furthermore, different codes are used on different runs. The initial runs on Martin's code were so fast that we decided to "size up" our problems, and went to APEX IV when it arrived on the Tech campus in September 1983. As we shall see, the runs continue to be fast, regardless of the formulation.

As we discuss the experiments in what follows, we will specify what formulations are used, whether or not simplications are made and what code is used.

We remark that, in the "expert systems" when implications like $K_i \rightarrow L_i$ occur, typically, K_i is a conjunction of literals and L_i is, in fact, a single literal, and it is rare for more than six or seven literals to occur in one implication. Such simple implications are of course of the generalized set covering variety. However, more complex implications can also occur.

Problem Generators. Each clause within both the Satisfiability and Implication Testing experiments is generated by randomly selecting

the s literals to appear in the clause. The literals are selected in the following manner. First, a random number, r, in (0,1.0) is selected, then, for the first literal of the clause, the interval (0, 1.0) is divided into ℓ equally-sized intervals. The interval in which r belongs determines the first letter of the clause. To determine the next letter a new r is generated and the interval is now divided into $(\ell-1)$ intervals, insuring that any one letter is never selected twice in the same clause. Once the clause consists of s literals, we determine whether each will appear as a letter, or the negation $(\sim A_i)$ of the letter, with equal probability.

For the <u>Satisfiability tests</u>, the first occurrence of any literal is randomly determined to be either a letter or its negation, each having equal probability of occurring. After the first occurrence, each subsequent occurrence is the opposite of the preceding logical type (i.e., if the first occurrence of A1 is as A1, the next is ~A1, the next A1). The selection technique is used to artificially make the problems difficult.

Once "c" independent clauses of the Satisfiability test are identified, each literal is scanned to determine the number of times it occurs. Any literal that appears only once, in one clause, is eliminated along with the entire clause. The scanning routine continues eliminating literals and clauses until each literal of the formula appears, as either a true literal or its negation, in at least two separate clauses. This last "pruning" step is to make the problems harder for humans to solve. After all, a letter which appears in only one clause can easily be given

a truth value to make that clause true, thus simplifying the problem. We note, however, that we had earlier done some trial runs without pruning, and this option (used below) does not seem to make any difference to machine solution.

The _Implication Testing_ experiment utilizes a different problem generator. The individual clauses (K_i, and L_i) are generated randomly as in the Satisfiability experiment. However, whether a literal occurs as a true literal or its negation is randomly determined for each occurrence. Each implication consists of "Λ" clauses on the left (K_i) and "V" clauses on the right (L_i). (e.g. (Al Λ A2) $\rightarrow$ (Al V A3).

Given the total number of literals present in the generated Implications problem, we fix a certain percentage of these letters to be either true or false. Then we test whether the overall system of implications and settings are consistent. However, to increase problem difficulty we require that at least one literal per implication be free (i.e. not fixed). In some tests we increase the number of free variables per implication to three.

Results

Satisfiability Experiment. The most significant result of our satisfiability experiments is the ease of their solution as mixed-integer programs. Tables 17 and 18 summarize our experiences using the BANDBX and Land-Powell's code, respectively. The actual problem sizes ranges from 114 constraints, 110 variables, 38 binary for the nonsharp modelling of Table 18, to 315 constraints, 132 variables 43 of which are binary in the ninth problem of Table 17. Problems sizes vary with the total number

of literal negations present, but for an approximate size use the following formulas.

	Constraints	Variables	Binary
Sharp	4C + L	2L + C	L
NonSharp	2C + L	2L + C	L

Aside from being extremely easy, even after our attempts to "harden" the problems, notice the sharp modellings are slower than the nonsharp modellings in nearly all instances. This is to be expected when the sharp formulation in this case is not better than the nonsharp formulation and yet has more constraints (recall that problems of this type are simply generalized set covering problems). The sharp modelling appears to perform worse than the nonsharp modelling as the number of literals per clause increases (Table 17). On the other hand, an increase in the number of clauses appears to affect the nonsharp modelling slightly more than the sharp (Table 18).

Table 17. Satisfiability Tests Using BANDBX.

| Problem Size | Satisfiable? | Total Nodes | | Time (CPU Seconds) | | | |
| | | | | Sharp | | Non Sharp | |
		Sharp	Non Sharp	LP	Total	LP	Total
L=31,C=44,S=2	No	2	2	9.9	11.8	4.0	6.5
L=35,C=45,S=2	No	3	3	12.8	21.5	4.7	9.8
L=37,C=45,S=2	Yes	1	4	8.7	8.8	4.7	10.8
L=36,C=52,S=2	No	3	3	16.0	27.0	6.0	14.4
L=46,C=63,S=2	No	2	3	25.7	29.8	8.7	21.3
L=53,C=68,S=2	Yes	Too Large	3	model too large		10.1	21.5
L=36,C=39,S=3	Yes	3	1	10.1	13.4	3.9	4.1
L=38,C=45,S=3	Yes	3	2	18.1	29.6	5.5	6.5
L=43,C=45,S=4	Yes	2	1	14.6	16.6	5.2	5.4
L=40,C=45,S=4	Yes	2	2	34.4	36.6	5.3	7.0
L=25,C=35,S=5	Yes	1	1	173.4	173.5	3.5	3.6

Table 18. Satisfiability Tests using Land-Powell's Code.

Problem Size	Satisfiable?	Total Nodes		Time (CPU Seconds)			
				Sharp		Non Sharp	
		Sharp	Non Sharp	LP	Total	LP	Total
L=38,C=40,S=2	Yes	2	1	8.4	9.7	3.4	3.5
L=42,C=40,S=2	Yes	1	1	7.1	7.3	3.6	3.7
L=44,C=45,S=2	Yes	2	2	9.7	12.6	4.8	6.5
L=45,C=45,S=2	Yes	2	5	9.1	11.0	4.7	13.0
L=44,C=52,S=2	Yes	1	1	12.5	12.6	5.5	5.6
L=43,C=55,S=3	No	2	3	14.7	17.1	6.3	13.7
L=45,C=60,S=3	No	2	3	19.3	22.0	7.6	13.4
L=45,C=60,S=4	No	2	3	17.7	23.9	7.3	15.5

Implications Testing Experiment. This experiment is performed entirely using CDC's APEX IV mixed-integer programming code, and as in the Satisfiability experiments, the branch-and-bound code, with few exceptions, solves these problems easily. The implications studied here are of the generalized set-covering type. In Tables 19 - 24 we do not apply our rule of having at least one free literal per implication, and the results show these random problems to be very easy. Recall that each implication requires only one generalized set covering constraint. Therefore the number of constraints in these problems equals the number of implications plus one constraint for each literal fixed at either true (=1) or false (=0).

Since we do not keep track of how many letters in a given clause are free (not fixed), it is possible that all letters (or all but one) in many clauses are fixed. Such problems tend to be very easy for humans to solve by a scanning procedure similar to our "pruning." Therefore in the next experiment we insure that the letters left free provided instances of NP-hard problems.

Table 20 summarizes problems in which there are five literals in each K_i and 2 in each L_i, an example is $(A1 \wedge \sim A3 \wedge A4 \wedge A5) \rightarrow (A2 \vee A3)$. Each implication must have at least three literals not assigned an initial value. This requirement greatly restricts the literals which we are able to assign as true or false. In the seventh example we are able to initialize only eleven of the fifty literals. In the "Total Fixed" column we indicate that each problem instance is solved twice. First the

Table 19. Implications Testing Using APEX IV.
No Free Variables Required.

| Problem Parameters | | | | | | | Time | |
| Total # of Implications | Literals | Literals/ Clause | | Total Fixed | Consistent ? | Nodes Total | APEX Units | |
		K	L				LP	Total
300	295	3	1	60	Yes	1	7.8	8.2
300	250	3	2	25	Yes	1	2.2	2.6
100	100	3	1	10	Yes	1	0.9	1.2
300	200	3	1	20	Yes	1	2.2	2.7
300	294	3	1	30	Yes	1	2.2	2.7
200	198	3	1	20	Yes	1	1.7	2.2
300	299	3	1	90	Yes	1	10.7	10.9
300	300	3	1	120	Yes	1	14.2	14.6
300	293	3	1	2	Yes	1	2.5	2.9
300	293	3	1	150	No	1	INFEAS	4.0
300	293	3	1	120	Yes	1	3.7	4.0
300	298	3	1	120	Yes	1	3.0	3.3
300	292	3	1	150	No	1	INFEAS	3.3
300	298	3	1	150	No	1	INFEAS	4.6
300	298	3	1	150	No	1	INFEAS	17.7

Table 20. Implication Tests Using APEX IV,
3 Free Literals per Implication

| Problem Parameters | | | | | | | Time | |
Total # of Implications	Literals	Literals/Clause K	L	Total Fixed	Consistent ?	Nodes Total	APEX Units LP	Total
300	160	5	2	58 to 1**	Yes	1	1.4	1.8
				58 to 0++	Yes	1	6.0	6.4
300	120	5	2	38 to 1	Yes	1	1.0	1.4
				38 to 0	Yes	2	5.1	5.9
300	100	5	2	34 to 1	Yes	1	1.3	1.7
				34 to 0	Yes	1	4.2	4.6
400	100	5	2	29 to 1	Yes	1	0.9	1.3
				29 to 0	Yes	1	4.4	4.8
400	60	5	2	13 to 1	Yes	1	1.1	1.4
				13 to 0	Yes	1	2.6	2.9
500	60	5	2	14 to 1	Yes	2	1.8	2.9
				14 to 0	Yes	3	32.5	47.5
500	50	5	2	11 to 1	Yes	1	1.2	1.6
				11 to 0	Yes	2	3.6	4.7
600	60	5	2	14 to 1	Yes	5	4.2	35.3
				14 to 0	Yes	1	25.3	25.7

*The APEX IV code was shut off before reaching a solution.

**This notation "15 to 1" means 15 literals are given as true (=1) and one is given as false (=0).

++This notation "15 to 0" means the same 15 literals which were previously given as true, one given as false (=0). Otherwise the problems within each set of lines are identical. In this case, there are actually 59 literals given as false (=0).

Table 21. Implications Tests, Affects of Fixing Literals,
1 Free Literal per Implication.

| Problem Parameters | | | | | | | Time | |
| Total # of Implications Literals | | Literals/Clause | | Total Fixed | Consistent ? | Nodes Total | APEX Units | |
		K	L				LP	Total
400	100	2	1	38	No	1	INFEAS	11.3
				37	No	1	INFEAS	10.9
				36	No	1	INFEAS	11.8
				34	No	1	INFEAS	16.5
				33	No	1	INFEAS	16.4
				32	No	1	INFEAS	17.4
				31	No	1	INFEAS	24.9
				30	No	1	INFEAS	28.1
				29	No	1	INFEAS	21.6
				28	No	1	INFEAS	22.5
				26	No	1	INFEAS	32.5
				24	No	1	INFEAS	31.5
				22	No	1	INFEAS	45.0
				20	No	1	INFEAS	40.9
				18	No	1	INFEAS	71.0
				16	No	1	INFEAS	62.4
				14	No	1	INFEAS	84.1
				12	No	13	88.7	266.8
				10	No	16	77.0	299.8
				8	Yes	13	75.6	170.2

(The empty Problem Parameters cells are annotated with the vertical text "SAME AS ABOVE".)

problem is solved with the literals assigned to be true, then the
identical problem is solved with the same literals assigned to be false
(=0). Since our random generator provides true and negation instances of
each literal with equal probability, whether a literal is assigned a true
or false value should not create more difficult problems. However, we
find that when literals are assigned false (=0) values, the problems are
more difficult to solve using the APEX IV code. Despite this
unexplainable phenomenon, the most prominent feature of these results is
their ease of solution. Again, what is easy or hard for humans may be
easy for machine solution.

In table 21 we solve only one problem instance, but vary the total
number of literals assigned a true value. Each implication consists of
only two literals in each K_i clause, and each L_i is a singleton clause.
The problem becomes more difficult as it approaches the feasible/
infeasible boundaries. In fact, we believe that some of the hardest
problems are at this boundary, and in these tests we are trying to create
a hard problem. Only the last three instances found the LP relaxation
feasible, and these three instances are much harder than the more
infeasible instances. The last entry actually found a feasible solution,
and its solution time is much faster than the infeasible example just
above it.

Using ideas from our experiment in connection with Table 21,
Tables 22 and 23 depict problems with fewer literals initially fixed.
These problems are difficult, one instance requires over 2000 APEX units
of time. (Recall, APEX units are virtually equal to cpu seconds.)

Table 22. Implication Tests Using APEX IV,
1 Free Literal per Implication.

| Problem Parameters | | | | | | | Time APEX Units | |
Total # of Implications	Literals	Literals/Clause K	L	Total Fixed	Consistent ?	Nodes Total	LP	Total
400	100	2	1	15 to 1**	No	3	46.4	71.8
400	100	2	1	15 to 1	No	1	INFEAS	63.2
				15 to 0++	No	7	135.2	193.9
400	100	2	1	15 to 1	No	21	77.7	237.6
				15 to 0	Yes	3	139.8	173.4
400	100	2	1	10 to 1	No	1	INFEAS	100.0
				10 to 0	No	1	INFEAS	147.7
400	100	2	1	10 to 1	No	1	INFEAS	84.7
				10 to 0	No	243	109.3	2083.5
400	100	2	1	5 to 1	Unknown	201*	74.4	557*
400	100	2	1	5 to 1	Unknown	481*	66.9	634*
400	100	2	1	1 to 1	Unknown	408*	44.0	716*

*The APEX IV code was shut off before reaching a solution.

**(see Table 20)

++(see Table 20)

Table 23. Implications Tests Using APEX IV
1 Free Literal per Implication.

Problem Parameters							Time	
Total # of Implications	Literals	Literals/Clause		Total Fixed	Consistent ?	Nodes Total	APEX Units	
		K	L				LP	Total
300	100	2	1	1 to 1	Yes	67	27.9	261.9
				1 to 0	Yes	8	21.2	53.0
300	100	2	1	1 to 1	Yes	32	25.7	184.2
				1 to 0	Yes	13	47.6	131.0
300	100	2	1	1 to 1	Yes	14	11.0	59.6
				1 to 0	Yes	16	13.8	92.9
300	100	2	1	15 to 1	Yes	12	15.9	51.3
				15 to 0	No	3	68.6	81.8
300	100	2	1	15 to 1	Yes	21	15.0	107.6
				15 to 0	No	1	INFEAS	69.4
300	100	2	1	15 to 1	Yes	8	19.1	61.3
				15 to 0	Yes	25	59.2	161.7
300	100	2	1	15 to 1	No	13	35.7	92.5
				15 to 0	No	83	64.6	431.0
300	100	2	1	15 to 1	Yes	7	38.6	80.1
				15 to 0	Yes	49	44.4	262.0
300	100	2	1	15 to 1	Yes	19	17.8	82.4
				15 to 0	Yes	6	39.0	57.4
300	100	2	1	15 to 1	No	2	29.7	35.6
				15 to 0	Yes	5	50.4	69.4
300	100	2	1	15 to 1	No	36	27.5	192.5
				15 to 0	Yes	88	93.8	570.9

Table 24. Implications Tests, Probability of Negations = 0.3, 1 Free Literal per Implication.

| Problem Parameters | | | | | | | Time | |
Total # of Implications	Literals	Literals/Clause K	L	Total Fixed	Consistent ?	Nodes Total	APEX Units LP	Total
300	100	2	1	15 to 1	Yes	10	28.9	64.6
				15 to 0	Yes	11	60.0	91.8
300	100	2	1	15 to 1	No	1	INFEAS	60.3
				15 to 0	Yes	10	68.6	102.6
300	100	2	1	15 to 1	Yes	4	40.3	61.0
				15 to 0	Yes	14	79.2	123.3
300	100	2	1	15 to 1	Yes	23	20.6	126.7
				15 to 0	No	3	79.0	88.7
300	100	2	1	15 to 1	No	1	INFEAS	33.8
				15 to 0	Yes	12	64.0	127.2
300	100	2	1	15 to 1	Yes	28	36.9	177.0
				15 to 0	Yes	16	104.9	164.8

In this manner, we succeed in producing problems which are hard to solve, apparently by being at the feasible/infeasible boundary. What seems hardest for the machine is when the LP is feasible, but barely, while the IP is infeasible. However, since these problems insure only one free letter per rule, they may have been easier for humans.

Again making literals false produces much more difficult problems, although in many cases these problems require much fewer branch-and-bound nodes. As a first attempt to explain why setting literals to false results in more difficult problems, we alter our problem generator in favor of generating true literals. In Table 24 the probability of a literal appearing as a negation is reduced to 0.3. For these problems we expect setting literals to be true to be even faster than before. Unfortunately that does not appear to be the case as 2 of the 6 problems actually solve faster when literals are forced to be false.

Table 25, 26, and 27 summarize the results of implications tests in which disjunctions occur in the clause preceding the implication. (i.e., for $K_i \rightarrow L_i$, both K_i and L_i are disjunctions of literals.) This test and corresponding integer model is outlined by formulae (5-15) to (5-17). We shall name model (5-16) as the STANDARD model, and model (5-17) as the SHARP model. For these tests we alter the problem generator such that each occurrence of a particular letter is exactly the opposite (True or False) of its previous occurrence. Thus, these problems instances are constructed to be difficult problems.

As (5-16) shows, the STANDARD model requires three constraints and one additional variable per implication. From (5-17), we see that the

Table 25. Implications with "ors" on LHS and RHS, 1 Free Literal per Implication, Variables Fixed at One.

Problem Parameters					Nodes Total		Time (APEX Units)			
Total # of Implications	Literals	Literals/Clause K	L	Total Fixed	SHARP	STAND	SHARP LP	SHARP Total	STANDARD LP	STANDARD Total
150	191	3	1	20	1	1	INF	13.478	INF	44.727
150	188	3	1	2	2	8	37.248	42.056	14.912	54.145
100	171	3	1	2	1	11	INF	8.282	6.783	29.522
150	194	3	2	20	3	49	10.283	14.933	34.487	209.627
150	149	3	2	20	1	1	INF	83.591	INF	85.357
170	150	3	2	20	1	1	INF	144.696	INF	165.928
100	188	3	2	2	1	16	4.639	5.08	4.240	22.361
150	199	3	2	2	1	24	7.566	8.039	5.064	39.089
150	196	3	2	2	4	40	7.469	14.334	1.157	150.869

Table 26. Implications with "ors" on LHS and RHS, 1 Free Literal per Implication, Variables Fixed at Zero.

Problem Parameters					Nodes Total		Time (APEX Units)			
Total # Implications	Literals	Literals/Clause K	Literals/Clause L	Total Fixed	SHARP	STAND	SHARP LP	SHARP Total	LP	STANDARD Total
150	191	3	1	20	1	1	INF	25.533	INF	79.09
150	188	3	1	2	1	3	INF	36.366	32.328	50.695
100	171	3	1	2	1	82	INF	13.267	7.673	93.548
150	194	3	2	20	11	37	69.919	114.417	86.692	181.684
150	149	3	2	20	3	190*	230.416	239.347	172.393	624*
170	150	3	2	20	12	485*	464.544	547.224	199.544	3,627**
100	188	3	2	2	1	13	5.06	5.501	4.157	14.397
150	199	3	2	2	3	13	12.175	19.333	10.268	37.30
150	196	3	2	2	6	426*	20.143	44.68	8.826	647*

*APEX IV was shut off before a solution was determined.

**Notice: after 1 hour of running time this problem remains unsolved using the STANDARD model.

SHARP model requires (S+1) constraints and an additional variable per implication, where S is defined as the number of letters appearing in the K_i disjunction. Based upon well-known results concerning constraint disaggregation, (see [19], [31], [47]), and our own previous experiments, we expect the SHARP formulation to outperform the STANDARD modelling despite its larger size. Our results support our expectation.

Tables 25 and 26 depict identical implication problems except in Table 25 the known letters are given to be true (set to 1) whereas in Table 26, the same letters are given as false (set to 0). The size of the implication problems of these tables result in modellings consisting of between 300 to 700 constraints and approximately 270-350 binary variables. Some problems involve a singleton on the right of the implication (L_i is a singleton) and others have a disjunction of two letters. Notice that in both (5-16) and (5-17) only one constraint is affected by adding disjunctions on the right of an implication.

For all problems in Table 25, the SHARP modelling overcomes its larger size and solves each instance faster and with fewer nodes. In one instance, (the third sample), the SHARP model finds the LP infeasible while the lack of sharpness allows the STANDARD to solve the LP and branch to 11 nodes before reaching the same conclusion. The node counts indicate at least a 4 to 1 reduction in favor of the SHARP model. For these samples, the SHARP LP is solved faster than the smaller, however more dense, STANDARD LP in 4 of the 9 instances.

Table 26 again indicates that setting the given variables to zero seems more difficult to APEX IV than setting the same variables to one.

Table 27. Implications with "ors" on LHS and RHS, 1 Free Literal per Implication, Variables Fixed at One.

| Problem Parameters | | | | | Nodes Total | | Time (APEX Units) | | | |
Total # Implications	Literals	Literals/Clause K	L	Total Fixed	SHARP	STAND	SHARP LP	SHARP Total	STANDARD LP	STANDARD Total
100	191	5	2	2	1	16	5.959	6.415	4.876	21.261
100	194	5	2	20	1	11	4.89	5.372	4.064	14.958
100	149	5	2	20	1	16	10.823	11.275	4.827	32.033
100	149	5	2	40	2	142	26.93	28.36	14.528	255.883
100	100	5	2	20	2	81*	64.24	71.586	9.509	347*

*Computer was stopped before finding problem infeasible.

Aside from that, the SHARP modelling is again faster and requires at most one-third the number of Branch-and-Bound nodes. For harder problems (see instances 5, 6, 9) the SHARP model displays at least a 15 to 1 advantage concerning node counts and in problem 6, the node comparison is at least 40 to 1. The actual time comparisons are unavailable as the STANDARD model fails to solve the instances after 5 minutes of cpu time, and still cannot solve problem 6 after one hour of cpu time (APEX units)!

In two inconsistent problems, the inconsistency is discovered with the SHARP LP, whereas the STANDARD modelling requires up to 82 branch-and-bound nodes before declaring one of them infeasible.

Table 27 summarizes problems with disjunctions of five letters on the LHS of each implication. Therefore each implication requires 6 constraints in the SHARP (5-17) modelling, and only three for the STANDARD (5-16) modelling. Even though the SHARP models now have twice the constraints, and their corresponding LP relaxations take longer to solve, they easily outperform the STANDARD formulation. The node count comparison reaches 70 to 1 in one instance and the total enumeration time advantage of the SHARP modelling increases as the problems become more difficult.

Summary

In general, both the SHARP and STANDARD models solve these logic problems very efficiently, and in most cases, very quickly. We do find an advantage of the SHARP modelling for implication/production rule problems in which disjunctions of letters occur before the implication $(K_i$ of $K_i \rightarrow L_i)$. This supports both ours and earlier efforts involving

constraint disaggregation. However our model comes automatically whereas previous disaggregations involve problem preprocessing. (see [31])

We are able to generate more difficult problems by exploring the feasible/infeasible boundaries. Whether these intentional efforts to create difficult problems is more or less realistic requires empirical work with actual expert systems. The results clearly indicate that these logic problems are very efficiently solved as math programs, especially considering that the mixed-integer branch-and-bound codes employed are not specialized binary variables codes, but general mixed-integer codes.

REFERENCES

1. E. Balas, "Disjunctive Programming and a Hierarchy of Relaxations for Discrete Optimization Problems," Carnegie-Mellon tech report, June 1983.

2. E. Balas, "Disjunctive Programming: Cutting-planes from Logical Conditions," in O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Nonlinear Programming 2, Academic Press, New York (1975), pp. 279-312.

3. E. Balas, "Disjunctive Programming: Facets of the Convex Hull of Feasible Points," no. 348, GSIA, Carnegie-Mellon University, 1974.

4. E. Balas and M. W. Padberg, "Set Partitioning: A Survey," SIAM Review 18 (1976), pp. 710-760.

5. Bazaraa, M. and Shetty, M., Non-Linear Programming, John Wiley & Sons, Inc., New York 1979.

6. Beale, E. M. L., "Branch-and-Bound Methods for Mathematical Programming," in Discrete Optimization II, eds. P. L. Hammer, E. L. Johnson, B. H. Korte, North-Holland Publishing Co., Amsterdam, pp. 201-221, 1979.

7. E. M. L. Beale and J. J. H. Forrest, "Global Optimization Using Special Ordered Sets," Mathematical Programming 10 (1976), pp. 52-69.

8. M. Benichou, J.-M. Gauthier, G. Hentges, G. Ribiere, "The Efficient Solution of Large-Scale Linear Programming Problems-Some Algorithmic Techniques and Computational Results," Mathematical Programming 13 (1977), pp. 280-322.

9. Bixby, Robert E., Matroids and Operations Research, Northwestern University, Illinois, 1984.

10. C. E. Blair and R. G. Jeroslow, "A Converse for Disjunctive Constraints," Journal of Optimization Theory and Its Applications 25 (1978), pp. 195-206.

11. S. Bradley, A. Hax and T. Magnanti, Applied Mathematical Programming, Addison-Wesley Pub. Co., Reading, Mass., 1977.

12. H. Crowder and E. L. Johnson, "Solving Large-Scale Zero-One Linear Programming Problems," Operations Research (1983), pp. 803-834.

13. G. B. Dantzig, Linear Programming and Extensions, Princeton, New Jersey, Princeton University Press, 1963.

14. R. Davis and D. Lenat, Knowledge-Based Systems in Artificial Intelligence, McGraw-Hill Book Co., New York, 1982.

15. R. Davis and B. Buchanan, "Production Rules as a Representation for a Knowledge-Based Consultation Program," Artificial Intelligience 8, (1977), pp. 15-45.

16. J. Edmonds, "Matroids and the Greedy Algorithm," Mathematical Programming 1 (1971), pp. 127-136.

17. G. D. Eppen and F. J. Gould, Quantitative Concepts for Management, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1979.

18. J.-M. Gauthier and G. Ribiere, "Experiments in Mixed-Integer Linear Programming Using Pseudo-Costs," Mathematical Programming 12 (1977), pp. 26-47.

19. A. M. Geoffrion and G. W. Graves, "Multicommodity Distribution System Design by Benders Decomposition," Management Science 20 (1974), pp. 822-844.

20. F. Glover, "New Results on Equivalent Integer Programming Formulations," Mathematical Programming 8 (1975), pp. 84-90.

21. F. Glover, "Polyhedral Annexation In Mixed Integer and Combinatorial Programming," Mathematical Programming 9 (1975), pp. 161-188.

22. R. E. Gomory, "An Algorithm for Integer Solutions to Linear Programs," in R. L. Graves and P. Wolfe, ed., Recent Advances in Mathematical Programming, McGraw-Hill, 1983.

23. A. C. Ho, "Cutting-planes for Disjunctive Programs: Balas' Aggregated Problem" Carnegie-Mellon University, October 1976 (a Ph. D. student summer research paper).

24. T. Ibaraki, "Integer Programming Formulation of Combinatorial Optimization Problems," Discrete Mathematics 16 (1976), pp. 39-52.

25. R. H. F. Jackson and R.P. O'Neill, eds. COAL Special Issue: Mixed Integer Programming in Mathematical Programming Systems, an ORSA and COAL/MPS publication.

26. R. Jeroslow, "Cutting-plane Theory: Disjunctive Methods," Annals of Discrete Mathematics 1 (1977), pp. 293-330.

27. R. Jeroslow, "Cutting-planes for Relaxations of Integer Programs," no. 347, GSIA, Carnegie-Mellon University, 1974.

28. R. Jeroslow, "Representations of Unbounded Optimizations as Integer Programs," Journal on Optimization Theory and Its Applications 30 (1980), pp. 339-351.

29. R. G. Jeroslow and J. K. Lowe, "Modelling with Integer Variables," Ga. Tech report (rev.), March 1983.

30. R. G. Jeroslow and J. K. Lowe, "Experimental Results on the New Techniques for Integer Programming Formulations," Ga. Tech report, June 1983.

31. E. L. Johnson, M. M. Kostreva, and U. Suhl, "Solving 0-1 Integer Programming Problems Arising from Large Scale Planning Models," IBM report RC9349, April 1982.

32. C. B. Krabek, "Some Experiments in Mixed Integer Matrix Reduction," Control Data Corporation, presented at ORSA/TIMS Hawaii Meeting, 1979.

33. A Land and S. Powell, Fortran Codes for Mathematical Programming: Linear, Quadratic and Discrete, John Wiley & Sons, London, England, 1973.

34. A. Land and S. Powell, "A Survey of Available Computer Codes to Solve Integer Linear Programming Problems," Research Report No. 81-9, London School of Economics and Political Science, London, April 1981.

35. E. Mendelson, Introduction to Mathematical Logic, D. Van Nostrand Co. Inc., Princeton.

36. R. R. Meyer, "Integer and Mixed Integer Programming Models: General Properties," Journal on Optimization Theory and its Applications 16 (1975), pp. 191-206.

37. R. R. Meyer, "Mixed-Integer Minimization Models for Piecewise-Linear Functions of a Single Variable," Discrete Mathematics 16 (1976), pp. 163-171.

38. R. R. Meyer, "A Theoretical and Computational Comparison of 'Equivalent' Mixed Integer Formulations," Naval Research Logistics Quarterly 28 (1981), pp. 115-131.

39. R. R. Meyer and M. V. Thakkar, "Rational Mixed Integer Minimization Models," MRC #1552, University of Wisconsin, 1976.

40. G. Mitra, "Investigation of Some Branch-and-Bound Strategies for the Solution of Mixed-integer Linear Programs," Mathematical Programming 4 (1973), pp. 155-170.

41. L. A. Oley and R. J. Sjoquist, "Automatic Reformulation of Mixed and Pure Integer Models to Reduce Solution Time in Apex IV," Control Data Corporation, presented at ORSA/TIMS San Diego Meeting October 1982.

42. R. Gary Parker and R. L. Rardin, Discrete Optimization, (in print) Ga. Inst. of Technology, Atlanta, 1984.

43. R. L. Rardin and U. Choe, "Tighter Relaxation of Fixed Charge Network Flow Problems," Report #J-79-18, Ga. Inst. of Technology, Atlanta, May 1979.

44. Rockafellar, R. T., Convex Analysis, Princeton University Press, 1970.

45. D. C. Sommer, "Computational Experience with the Ophelie Mixed Integer Code," Control Data Corporation, presented at ORSA/TIMS, 1970.

46. Stoer, J., and Witzgall, C., Convexity and Optimization in Finite Deimensions I, Springer-Verlag, New York, 1970.

47. H. P. Williams, "Experiments in the Formulation of Integer Programming Problems," Mathematical Programming Study 2, 1974, pp. 180-197.

48. H. P. Williams, Model Building in Mathematical Programming, John Wiley and Sons (Wiley Interscience), 1978.

END

FILMED

D